ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN

DEPARTAMENTO DE INGENIERÍA AUDIOVISUAL Y TELECOMUNICACIÓN

CORRELACIÓN ENTRE SEÑALES DE AUDIO MULTI-MICROFÓNICAS

Realizado por:

Jesús Javier Alonso Reguilón

Supervisado por:

Antonio Pedrero González

Tema: Alinemiento de tomas multi-microfónicas

Título: Correlación entre señales de audio multi-microfónicas

Autor: Jesús Javier Alonso Reguilón

Titulación: Ingeniería Técnica de Telecomunicaciones, especialidad Sonido e Imagen

Tutor: Antonio Pedrero González

Departamento: Ingeniería Audiovisual y Comunicaciones

Tribunal

Presidente: Jose Antonio Sánchez Fernández

Vocal: Antonio Pedrero González

Vocal Secretario: Francisco Javier Tabernero Gil

Fecha de Lectura: 9 de septiembre de 2013

A mi familia, que siempre han estado a mi lado apoyándome en todo momento.

A mis amigos y compañeros por la motivación para seguir en esto del sonido.

Índice general

Capítulo 0

Resumen

0.1. Resumen

Este proyecto pretende mostrar los desfases existentes entre señales de audio obtenidas de la misma fuente en distintos puntos distanciados entre sí. Para ello nos basamos en el análisis de la correlación de las señales de audio multi-microfónicas, para determinar los retrasos entre dichas señales .
Durante las de tres partes diferentes que conforman este proyecto, explicaremos el dónde, cómo y por qué se produce este efecto en este tipo de señales.

En la primera se presentan algunos de los conceptos teóricos necesarios para entender el desarrollo posterior, tales como la coherencia y correlación entre señales, los retardos de fase y la importancia del micro-tiempo. Además se explican diversas técnicas microfónicas que se utilizarán en la tercera parte.

A lo largo de la segunda, se presenta el software desarrollado para determinar y corregir el retraso entre las señales que se deseen analizar. Para ello se ha escogido la herramienta de programación Matlab, ya que ha sido la más utilizada en la mayoría de las asignaturas que componen la titulación y por ello se posee el suficiente dominio de la misma. Además de presentar el propio software, al final de esta parte hay un manual de usuario del mismo, en el que se explica el manejo para posibles usos futuros por parte de otras personas interesadas.

En la última parte se demuestra en varios casos reales, el estudio de la alineación de tomas multi-microfónicas en las cuales se produce en efecto que se intenta detectar y corregir. Aquí se realizan tres estudios de dicho fenómeno.
En el primero se emplean señales digitales internas, concretamente ruido blanco, retrasando algunas muestras dichas señales unas de otras, para luego analizarlas con el

software desarrollado y comprobar la eficacia del mismo.
En el segundo se analizan la señales de audio obtenidas en el estudio de grabación de varios grupos de música moderna, mostrando los resultados del empleo del software en algunas de ellas, tales como las tomas de batería, bajo y guitarra.
En el tercero se analizan las señales de audio obtenidas fuera del estudio de grabación, en donde no se dispone de las supuestas condiciones ideales que se tienen en el entorno que rodea a un estudio de grabación (acústicamente hablando). Se utilizan algunas de las técnicas microfónicas explicadas en el último apartado de la parte dedicada a los conceptos teóricos, para la grabación de una orquesta sinfónica, para luego analizar el efecto buscado mediante nuestro software, presentando los resultados obtenidos. De igual manera se realiza en el estudio con una agrupación coral de cuatro voces dentro de una Iglesia.

0.2. Abstract

This project aims to show delays between audio signals obtained from the same source at different points spaced apart. To do this we rely on the analysis of the correlation of multi-microphonic audio signals, to determine the delay between these signals.
During three different parts that make up this project, we will explain where, how and why this effect occurs in this type of signals.

At the first part we present some of the theoretical concepts necessary to understand the subsequent development, such as coherence and correlation between signals, phase delays and the importance of micro-time. Also explains several microphone techniques to be used in the third part.

During the second, it presents the software developed to determine and correct the delay between the signals that are desired to analyze. For this we have chosen the programming software Matlab , as it has been the most used in the majority of the subjects in the degree and therefore has sufficient command of it. Besides presenting the software at the end of this part there is a user manual of it , which explains the handling for future use by other interested people.

The last part is shown in several real cases, the study of aligning multi- microphonic sockets in which it is produced in effect trying to detect and correct. This includes three studies of this phenomenon.
In the first internal digital signals are used, basically white noise, delaying some samples the signals from each other, then with software developed analyzing and verifying its effectiveness.
In the second analyzes the audio signals obtained in the recording studio several contemporary bands, showing the results of using the software in some of them, such as the taking of drums, bass and guitar.
In the third analyzes audio signals obtained outside the recording studio, where there are no ideal conditions alleged to have on the environment surrounding a recording studio (acoustically speaking). We use some of the microphone techniques explained in the last paragraph of the section on theoretical concepts, for the recording of a symphony orchestra, and then analyze the effect sought by our software, presenting the results. Similarly, in the study performed with a four-voice choir in a church.

Capítulo 1

Introducción

En los seres humanos existen mecanismos que interpretan las señales acústicas que se producen a nuestro alrededor. Por ejemplo, percibimos la distancia a la que se encuentra un león cuando ruge y esto nos avisa del posible peligro que alberga esperar a su llegada y el tiempo de huída del que disponemos (si es que éste aún existiera).

Cuando escuchamos por ejemplo una orquesta sinfónica nuestro cerebro también percibe la distancia a la que se encuentran los instrumentos y los acentos que éstos realizan durante la interpretación de la propia obra (siempre bajo la mirada subjetiva del director quien dicta dónde, cuándo y cómo los acentos deben realizarse).

El problema se presenta cuando tratamos de plasmar las interpretaciones de esa orquesta para que no caigan en el olvido, y realizamos una grabación de la misma empleando varios micrófonos en la misma. Aquí cabe plantearse la primera cuestión, ¿llegarán al mismo tiempo las señales de las diferentes zonas de la orquesta recogidas por los micrófonos repartidos que al micrófono situado sobre la cabeza del director? y en ese caso surgiría la segunda cuestión, ¿tendría ello alguna repercusión sobre la percepción auditiva del ser humano de dicha grabación?

Estas cuestiones son la que se intentará dar una respuesta con el desarrollo de este proyecto, extrapolando el ejemplo de la grabación de la orquesta sinfónica a todas las situación en las que se requiera grabar una fuente o conjunto de fuentes de manera multi-microfónica, es decir, captando dichas fuentes de manera simultánea por varios micrófonos repartidos por todo el espacio de acción de la fuente.

Capítulo 2

Conceptos teóricos

2.1. Micro-Tiempo

Para explicar la importancia del micro-tiempo vamos a estudiar y comparar las siguientes escalas:

- La escala de tiempo de la frecuencia del audio digital.
- La escala de tiempo de la propagación del sonido en el aire.
- La escala de tiempo musical, es decir, el pulso o el ritmo.

Y de la misma forma las unidades de tiempo que vamos a estudiar son:

- 1 muestra
- 1 milisegundo.

-Escala de tiempo: frecuencia de muestreo.

Cuadro 2.1: La duración de una muestra depende de la frecuencia de muestreo

Frecuencia de muestreo	Muestras/ms	1 duración de muestra
32 KHz	32	31.25 μs
44.1 KHz	44.1	22.68 μs
48 KHz	48	20.83 μs
88.2 KHz	88.2	11.34 μs
96 KHz	96	10.42 μs
192 KHz	192	5.21 μs

En un milisegundo (el periodo de una onda de 1 kHz) pasa de 32 a 192 muestras.

-Escala de tiempo: propagación del sonido

Teniendo en cuenta que la velocidad de propagación del sonido en el aire es alrededor de 340 m/s, en un milisegundo, una onda recorrerá 34 cm. Esto perjudica seriamente a señales muy coherentes porque dependiendo de la frecuencia se sumarán o cancelarán. En este caso, una señal de 500,1500,2500,3500, etc ... Hz se cancelará, mientras que

una señal de 1000,2000,3000, etc ... Hz se amplificará. Esto es conocido como el efecto de filtro peine, más conocido por su denominación en inglés "comb filter" (hablaremos sobre él más tarde).

Este efecto es usado artísticamente en guitarras y otros instrumentos recibiendo el nombre de *flanging, phasing,* etc. Sin embargo, este efecto es perjudicial en otros muchos entornos que son el objeto del estudio en este proyecto. Ejemplos de ello son un locutor hablando en un atril con dos micrófonos, una guitarra captada a través de dos micrófonos separados entre ellos, etc.

Respecto a las otras unidades de tiempo, el retraso o desfase de 1 muestra no tiene ningún efecto audible, apreciándose cuanto más grande sea este retraso o desfase.

-Escala de tiempo: musical (ritmo/pulso).

Para esta sección, consideramos un tempo Allegro, aproximadamente 120 bpm (*beats per minute*). Asociamos estos "beats" (pulsos) con una figura musical, por ejemplo la figura negra. De este modo, significa que tocaremos 120 negras por minuto. Sabemos que la duración de las figuras musicales se estructura como muestra la figura 2.1. Podemos calcular el tiempo de cada figura utilizando la fórmula 2.1. Y como podemos observar en la figura 2.2, la duración de tiempo del resto de las figuras.

$$Tfigura = \frac{60 segundos}{1 min} \frac{1 min}{bpm} \tag{2.1}$$

Un milisegundo es mucho menos que la duración de una semifusa. Un retraso de un milisegundo en la ejecución de una nota no tiene relevancia (tanto por ejecución humana por por un secuenciador). Además,un retraso de una muestra de audio resulta despreciable, mientras que un retraso de 40 milisegundos es notable para la mayoría de los músicos profesionales.

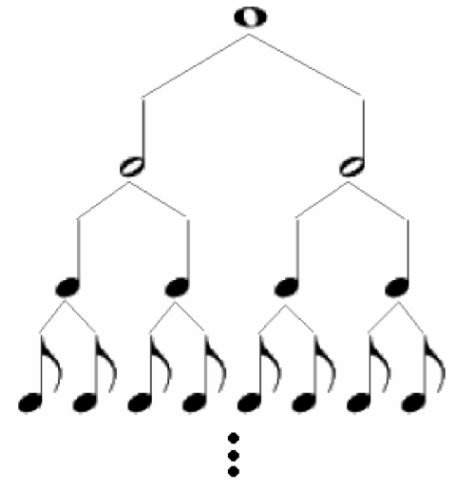


Figura 2.1: Notación de las notas musicales.

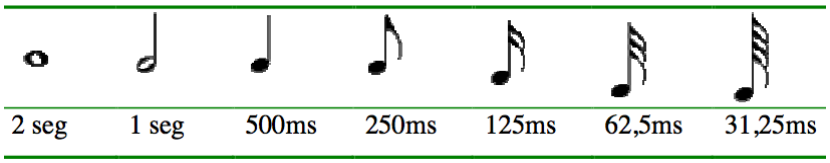


Figura 2.2: Duración de las notas musicales en milisegundos.

2.2. Coherencia y correlación

La coherencia se define como: diferencia de fase constante entre dos o más ondas a lo largo del tiempo. Dos ondas se dice que están en fase si sus máximos y mínimos coinciden en en mismo lugar y al mismo tiempo respectivamente, y estarán fuera de fase si no coinciden. Las ondas serán incoherentes si los máximos y mínimos se encuentran aleatoriamente.

La correlación se define como: medida de similitud entre dos cantidades (vibración de forma de ondas o señales), en otras palabras, es una medida de la similitud de dos formas de onda como una función de intervalos de tiempo aplicada a cada uno de ellos. Si aplicamos este concepto al procesado de sonido, podemos usar la correlación cruzada para comparar dos o más señales.
El resultado es un valor que oscila entre -1 y 1, donde 1 indica que las dos señales tienen una relación lineal positiva (en otras palabras, sus intervalos tienen siempre la misma polaridad). Un coeficiente de correlación igual a -1 indica que las dos señales se relacionan lineal y negativamente (además, sus intervalos siempre tienen polaridades opuestas). En caso de señales de banda ancha, una correlación de 0 normalmente indica que las dos señales están completamente separadas o separadas en tiempo por un retraso mayor que la media de tiempo.

2.3. Coloración

La coloración se define como los cambios en el timbre de la señal de audio. Añadiendo una reflexión cambia automáticamente la respuesta de frecuencia de una señal, dando algún tipo de coloración. Esto podría ser visto como una distorsión, producida por el efecto de filtro peine comentado anteriormente.
Se trata de un sonido hueco “silbante”, que resulta de la cancelación de fase entre

múltiples micrófonos. Cuando una sola voz es recogida por dos micrófonos diferentes separados ligeramente, el sonido de la voz en el micrófono más distante se "retrasa" en relación con el micrófono más cercano. La mezcla de estas dos señales hará que la cancelación a ciertas frecuencias cree una señal respuesta que se parece a un filtro de peine invertida.

2.4. Alineamiento de tiempo y fase.

La fase de una onda de sonido de una única frecuencia siempre se describe en relación con el punto de partida de la onda o 0 grados (figura 2.3).El cambio de presión también es cero en este punto. El pico de la zona de alta presión está a 90 grados, y el cambio de la presión cae de nuevo a cero a 180 grados. El pico de la zona de baja presión está a 270 grados, y el cambio de presión se eleva a cero a 360 grados para el inicio del siguiente ciclo.

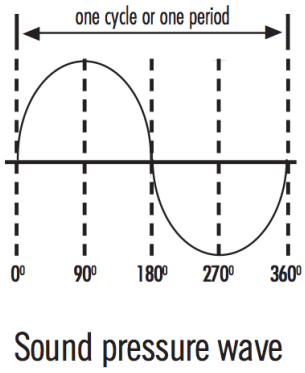


Figura 2.3: Fase de un tono puro.

Dos ondas sonoras idénticas que comienzan en el mismo punto en el tiempo se dice que están "en fase" (figura 2.4) y se sumarán creando así una única onda con el doble de amplitud pero por lo demás idéntica a las ondas originales. Dos ondas sonoras idénticas

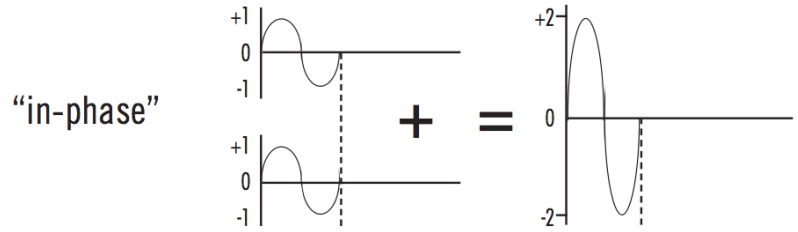


Figura 2.4: Frecuencia de un tono puro .[en] fase”.

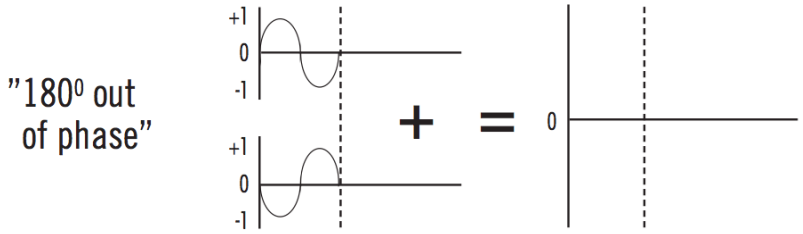


Figura 2.5: Frecuencia de un tono puro ”fuera de fase”.

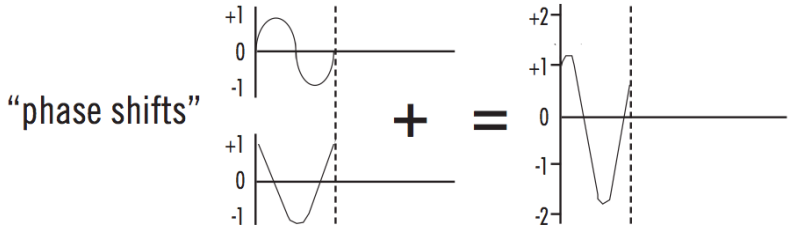


Figura 2.6: Frecuencia de un tono puro con desplazamiento de fase

con el punto de partida de una de ellas a 180 grados de la otra onda se dice que están en “oposición de fase”(figura 2.5), y las dos ondas se anularán entre sí completamente. Cuando se combinan dos ondas de sonido de la misma frecuencia pero parten de diferentes puntos en el tiempo, la onda resultante se dice que tiene “desplazamiento de fase”(figura 2.6). Esta nueva onda tendrá la misma frecuencia que las ondas originales pero habrá aumentado o disminuido la amplitud en función del grado de diferencia de fase. El cambio de fase, en este caso, indica que los 0 grados de dos ondas idénticas no son los mismos.

La mayoría de las ondas de sonido no están formadas por una única frecuencia, sino que se componen de muchas frecuencias. Cuando idénticas ondas sonoras de frecuencias múltiples se combinan, hay tres posibilidades para la onda resultante: la duplicación de la amplitud en todas las frecuencias, si las ondas están "en fase"; una cancelación completa en todas las frecuencias, si las ondas están en "oposición de fase"; o cancelación parcial y el refuerzo parcial en varias frecuencias, si las ondas tienen relación de fase intermedia.

El último caso es el más probable, y el resultado audible es una respuesta en frecuencia degrada, denominada filtro de peine (figura 2.7). El patrón de picos y valles se asemeja a los dientes de un peine, y la profundidad y ubicación de estas muescas dependerá del grado de desplazamiento de fase.

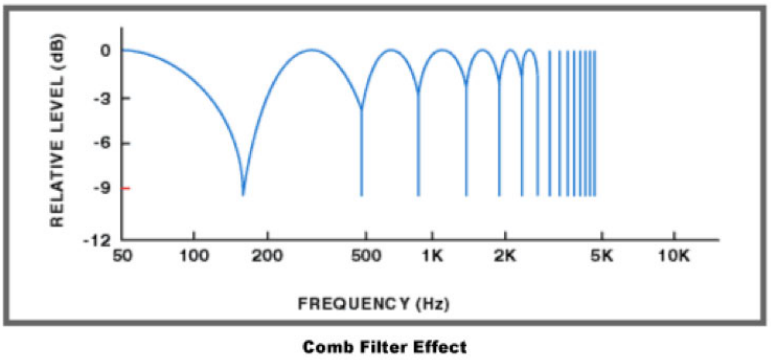


Figura 2.7: Filtro peine producido por la llegada de sonido a dos micrófonos separados cierta distancia.

En la utilización de micrófonos, este efecto puede producirse cuando dos (o más) micrófonos captan la misma fuente de sonido a diferentes distancias. Debido a que el sonido tarda más tiempo en llegar al micrófono que situado a mayor distancia, existe efectivamente una diferencia de fase entre las señales de los micrófonos cuando se com-

binan eléctricamente en el mezclador. El filtro peine resultante depende de la diferencia de tiempo de llegada del sonido entre los micrófonos: una gran diferencia de tiempo (amplia distancia entre micrófonos) causa que filtro peine comience a bajas frecuencias, mientras que una pequeña diferencia de tiempo (corta distancia entre micrófonos) mueve el filtro peine hacia frecuencias más altas.

2.4.1. Alineación en la tecnología del sonido.

Hay escenarios en los que la alineación del tiempo y la coherencia son de gran importancia. Especialmente cuando queremos lograr un sonido mejor, más claro y bien dirigido. Antes en la época analógica esto era imposible de lograr. Fue gracias a la tecnología digital de audio cuando la corrección de la coherencia y de la alineación de temporal se hizo posible. Hoy en día, todas las mesas de mezclas digitales y DAWs tienen compensación de retardo al alcance de tus manos. Ajustar retrasos puede ser una tarea que consume tiempo ya que implica medir distancias entre las fuentes de sonido y la colocación del micrófono con especial cuidado.

El problema de la contaminación acústica (explicado más adelante) es tan antiguo como la tecnología de sonido, y para aliviarla se han utilizado varias técnicas, algunas de ellas son:

- Aproximación de los sensores (micrófonos) a las fuentes para mejorar la relación deseada de la fuente de sonido con respecto a las fuentes no deseadas.
- Utilizando diagramas polares de captación. Por desgracia, las curvas polares de hoy en día siguen siendo de primer orden (omnidireccional, cardioide, hyper-cardioide, y figura de ocho), excepto los cañones y las parábolas que son un tímido intento de mejorar la directividad.

- Separación de las fuentes sonoras, o su aislamiento cuando sea posible

- Minimizar el impacto del refuerzo sonoro.Por ejemplo, monitorizar mediante audífonos *in earl* en lugar de utilizar las cuñas de escenario.

Ahora, es el momento de la alineación. Cuando se han utilizado todas las técnicas anteriores y no pueden dar más de sí, entonces viene el concepto de alineación temporal para ayudarnos. El concepto es simple. Si una fuente se va a capturar mediante más de un micrófono, tenemos que intentar que los sensores (micrófonos) estén compensados mediante retrasos para que la captación de la fuente contaminante coincida en el tiempo, es decir, que están alineados en el tiempo.

2.5. Métodos microfónicos de grabación

La mayoría de los ingenieros que hacen grabaciones comerciales utilizan matrices mixtas en las que una pareja central, ya sea coincidente o casi-coincidente, se combina con un par de micrófonos omnidireccionales para los laterales. La combinación proporciona una excelente flexibilidad y permite al ingeniero alterar perspectivas en la orquesta sin necesidad de hacer ajustes en las posiciones de los propios micrófonos.

2.5.1. Técnicas esteroefónicas

Las técnicas microfónicas en estéreo se utilizan principalmente para grabar música clásica, agrupaciones y solistas sobre el terreno. Estos métodos capturan un evento de sonido en su conjunto, por lo general usando sólo dos o tres micrófonos. Durante la reproducción de una grabación estéreo, las imágenes de los instrumentos musicales se escuchan en varios lugares entre los altavoces estéreo. Estas imágenes sitúan a los instrumentos en los mismos lugares, de izquierda a derecha, en donde estaban durante

la sesión de grabación. Además, la microfonía estéreo transmite.

- La profundidad o distancia de cada instrumento.
- La distancia de la agrupación respecto del oyente (la perspectiva).
- La sensación espacial del entorno acústico, la reverberación del ambiente o la sala.

Las tres técnicas estereofónicas más comunes son:

1.- Par coincidente

Par XY

La configuración XY más utilizada consta de dos micrófonos cardioides de primer orden orientados típicamente a 90 ° para producir una imagen estéreo (figura 2.8). Teóricamente, las dos cápsulas de micrófono tienen que estar exactamente en el mismo punto para evitar cualquier problema de fase debido a la distancia entre las cápsulas.

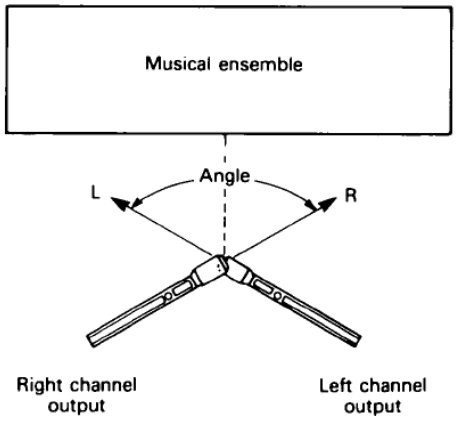


Figura 2.8: Configuración par XY.

Como esto no es posible, la mejor aproximación a la colocación de dos micrófonos en el mismo punto es poner un micrófono encima del otro con los diafragmas alineados

verticalmente. De esta manera, las fuentes de sonido en el plano horizontal serán recogidos como si los dos micrófonos se colocan en el mismo punto.

La imagen estéreo se produce por la atenuación fuera del eje de los micrófonos cardioides. Mientras la técnica estéreo AB (apartado Par AB 2.5.1) consiste en una diferencia de tiempo, el Par XY es una diferencia en el nivel. Pero como la atenuación fuera del eje de un micrófono cardioide de primer orden es sólo 6 dB en 90 °, la separación entre canales es limitada, por lo que no se percibe una gran diferencia de las imágenes estereofónicas con este método de grabación. Por lo tanto, XY estéreo se utiliza a menudo cuando se necesita una alta compatibilidad mono - por ejemplo, en situaciones de radiodifusión donde muchos oyentes siguen recibiendo el audio en equipos mono.

Par MS

Una forma especial de la técnica de par coincidente es la Mid-Side (MS) ilustrada en la Figura 2.9. Utiliza 2 micrófonos diferentes y una matriz especial para crear la propagación estéreo y la localización. La matriz MS utiliza las señales de fase entre el micrófono central y el micrófono ambiental para producir una señal de izquierda a derecha adecuada para un equipo de música convencional.

Debido a la presencia del micrófono central, esta técnica es muy adecuada para grabaciones estéreo donde se necesita una buena compatibilidad monofónicos, y es bastante popular en la radiodifusión.

La forma más fácil de entender la matriz MS es mediante el estudio de estos dos cálculos simples: Canal izquierdo = M + S

Canal derecho = M - S

El sistema MS dará información direccional desde el micrófono bidireccional que captura en dos direcciones y sólo muestra una señal con polaridad opuesta. Cuando el sonido se acerca a la matriz MS desde la derecha, entra en el micrófono bi-direccional lateral en su lado de fase invertida. El cálculo matricial dice que la salida correcta es

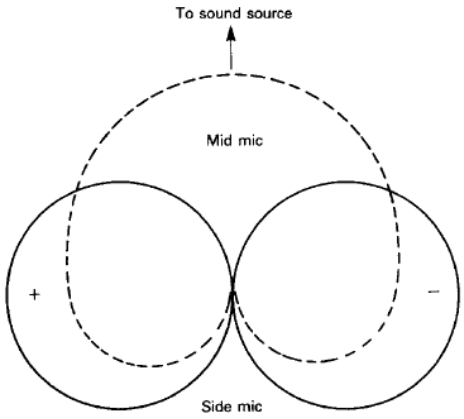


Figura 2.9: Configuración Mid-side. Canal izquierdo = central (M) + ambiental (S). Canal derecho = central (M) – ambiental (S). La polaridad de los lóbulos secundarios del micrófono ambiental (S) se indica mediante los signos más y menos.

M menos S. Si resta algo a una señal a partir de otra invertida de fase es como la suma de ésta. Matemáticamente hablando negativo por negativo es positivo. Por lo tanto, aquí está, el sonido al acercarse a la matriz MS desde el lado derecho crea una señal en la salida derecha de la matriz debido a una resta simple.

Par Blumlein

La configuración del par Blumlein utiliza dos micrófonos bidireccionales en el mismo punto (concidentes) con un ángulo de 90 ° entre sí (figura 2.10). Esta técnica estéreo normalmente dará los mejores resultados cuando se utiliza a distancias relativamente cortas de la fuente de sonido, ya que como micrófonos bidireccionales utilizan la tecnología de transductor de gradiente de presión y por lo tanto se encuentran bajo la influencia del efecto de proximidad.

A distancias mayores, estos micrófonos perderán las frecuencias graves. El estéreo Blumlein recoge información relacionada puramente con la intensidad. Tiene una sepa-

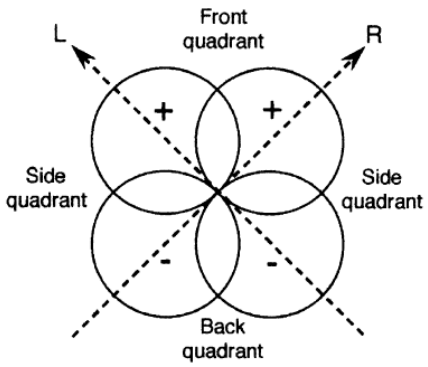


Figura 2.10: Configuración del par coincidente Blumlein.

ración de canal más alta que el estéreo XY, pero tiene la desventaja de que las fuentes de sonido situadas detrás del par estéreo también se recogerán y reproducirán con la fase invertida.

2.- Par espaciado

Par A-B

La Técnica del par A-B utiliza dos micrófonos separados (a menudo omnidireccionales) para grabar señales de audio. La separación entre micrófonos introduce pequeñas diferencias en la información de tiempo o fase contenida en las señales de audio (de acuerdo con las direcciones relativas de las fuentes de sonido). De igual manera que el oído humano puede detectar diferencias de tiempo y fase en las señales de audio y utilizarlas para la localización, las diferencias de tiempo y fase actuarán como señales estéreo para permitir a la audiencia "capturar el espacio" en la grabación, y experimentar una potente y clara imagen estéreo de el campo sonoro completo, incluyendo el posicionamiento de cada fuente de sonido individual y los límites espaciales de la propia sala.

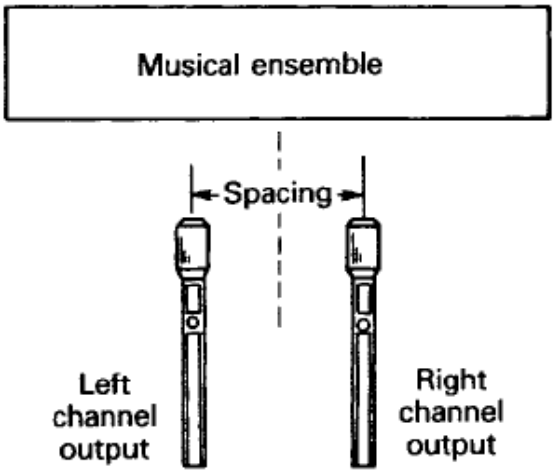


Figura 2.11: Spaced pair set-up.

Una consideración importante al establecer una configuración para las grabaciones estéreo A-B, es la distancia entre los dos micrófonos. Usando una separación recomendada entre micrófonos de un cuarto de la longitud de onda del tono más grave, y teniendo en cuenta la limitada capacidad del oído humano para localizar frecuencias por debajo de 150 Hz, nos conduce a una separación óptima entre micrófonos de aproximadamente entre 40-60 cm. Las separaciones pequeñas entre micrófonos se utilizan a menudo cerca de la fuente sonora para evitar que la imagen de sonido de un instrumento musical en particular sea "demasiado amplia" y poco natural. Los espaciados por debajo de 17 a 20 cm son detectables por el oído humano, ya que esta distancia es equivalente a la distancia entre los oídos.

Los micrófonos espaciados plantean problema. Las grandes diferencias de tiempo entre los canales corresponden a las bastas diferencias de fase entre canales. En oposición de fase, las señales de baja frecuencia pueden causar excesiva modulación vertical de un ritmo musical. Además, la combinación de ambos micrófonos en monofónico a veces provoca cancelaciones de fase a diferentes frecuencias, que pueden ser audibles o no.

Decca tree

Introducido originalmente por el sello discográfico Decca [1], el "árbol"se compone de tres puntos de recogida donde se encuentran los micrófonos omnidireccionales formando un triángulo (a menudo equilátero) apuntando a la fuente de sonido. Los dos micrófonos externos están tan lejos que se producirá un .ªgujero en el centro"por lo que se hace necesario introducir un micrófono en el centro. Éste debe ser mezclado para llenar este agujero, con cuidado de no hacer que la perspectiva de sonido se vuelva demasiado monofónica.

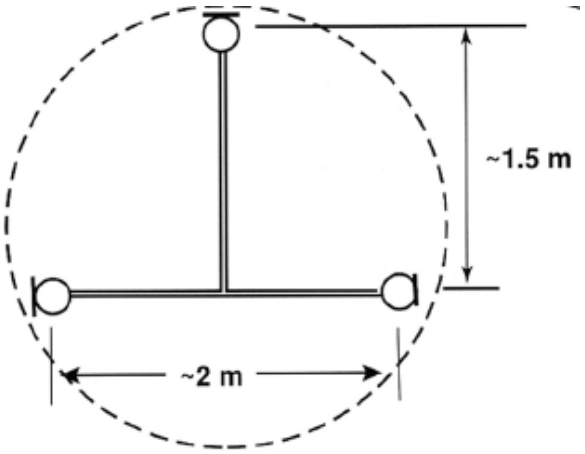


Figura 2.12: Configuración del "árbol Decca" (Decca tree).

Las distancias exteriores oscilan entre 60cm a 120cm. El tercero, en el centro, puede estar ligeramente por debajo y delante de la par exterior. Dependiendo de las variables acústicas de la sala donde se está grabando el conjunto o la orquesta, el árbol puede elevarse o bajarse para lograr un mejor resultado. Se trata de un estilo de ubicación de un gran éxito, ya que garantiza un sonido natural, sin fisuras para la audiencia, y les permite experimentar la obra musical en todo su contexto de dinámica global. A menudo, el árbol se coloca justo detrás o sobre el director, lo que da lugar a un balance

de sonido muy próximo a la intención de éste. Los 3 micrófonos, además, se acercan más a las secciones de orquesta que las configuraciones de AB, proporcionando aún más claridad y una proyección más nítida de la imagen sonora estereofónica.

3.- Par casi-coincidente

Como se muestra en la Figura 2.13, la técnica estereofónica casi-coincidente utiliza dos micrófonos orientados con un cierto ángulo, y con sus rejillas espaciadas horizontalmente unos pocos centímetros de distancia. Incluso unos pocos centímetros de distancia aumenta la propagación estéreo y añade sensación de profundidad y ligereza a la grabación. Cuanto mayor es el ángulo o la separación entre los micrófonos, la mayor es la percepción estéreo.

Así funciona este método: la angulación entre los micrófonos produce diferencias de ni-

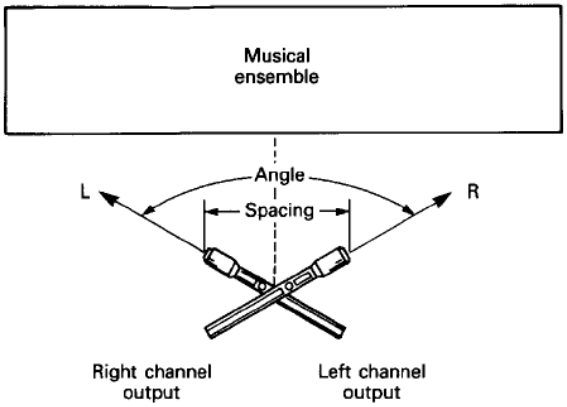


Figura 2.13: Configuración del par casi-coincidente.

vel entre canales, y la distancia produce diferencias de tiempo. Las diferencias de nivel entre canales y las diferencias de tiempo se combinan para crear el efecto estéreo. Si la orientación o la distancia es demasiado grande, el resultado es separación exagerada de la imagen estéreo. Si la orientación o la distancia es demasiado pequeña, el resultado es una separación de la imagen estéreo estrecha.

El ejemplo más común del método casi-coincidente es la técnica ORTF (Oficina de Radiodifusión y Televisión Francesa, Organización francesa de radiodifusión a la que se le acredita el desarrollo de esta técnica), que utiliza dos cardioides en ángulo de inclusión de 110 º y espaciados 17 centímetros en sentido horizontal . Este método tiende a proporcionar la localización exacta, es decir, los instrumentos a los lados de la orquesta se reproducen muy cerca o incluso localizados en los propios altavoces, y los instrumentos a mitad de camino entre el centro y un lado tienden a ser reproducido a mitad de camino entre el centro y un lado.

Capítulo 3

Estudio, desarrollo y manejo del software

3.1. Alineamiento entre señales de audio

El problema de la incoherencia del sonido está constantemente presente en el momento en que mezclamos dos o más micrófonos en el mismo escenario de sonido. Hay un montón de ejemplos al respecto, pero algunos de los más destacables pueden ser:

(A) Varios interlocutores cercanos cada uno con micrófono de solapa.

(B) Grabación multi-microfónica de una orquesta (sin refuerzo sonoro) con un par estéreo y varios micrófonos de apoyo repartidos por la escena.

(C) Micrófonos de toma de ambiente (audiencia) en un lugar con o sin refuerzo electroacústico.

Este efecto es lo que se puede denominar *contaminación acústica*.
La tecnología de sonido (como mencionamos en la sección 2.4.1) se basa fundamentalmente en disponer de micrófonos que tienen que captar fuentes. Esto es fácil de entender, pero los problemas surgen cuando tenemos varias fuentes de sonido próximas entre sí o una es dominante, es decir, más fuerte que el resto. Los micrófonos recogen lo que se quiere (que es la fuente), y lo que no se quiere (fuentes vecinas). Puede parecer que esto no es un problema, pero haciendo referencia al tema *el micro-tiempo* antes mencionado en la sección 2.1, se observa que la velocidad de propagación del sonido es muy lenta unido a que las características de percepción del oído humano hacen que la suma de una señal consigo misma mínimamente retrasada hace que el resultado sea extraño a nuestro cerebro.
Esta cualidad de la percepción humana (relacionado con el efecto *Haas*) es lo que utilizamos para localizar espacialmente las fuentes sonoras. Por lo tanto, los retrasos en el rango de micro-tiempo es una poderosa herramienta que realmente puede conseguir un sonido a la altura de la época actual, si aprendemos a manejarlo.

3.1.1. Aplicaciones en la música.

Las aplicaciones dentro del mundo de la música pueden ser muy diversas, tanto en la música clásica como la música más moderna. Dentro del entorno de la música clásica, podría enfocarse hacía la grabación de orquestas. La gran mayoría de estas grabaciones se llevan a cabo mediante *arrays* de micrófonos distribuidos meticulosamente por todos los instrumentos de la orquesta para tener una referencia de todas las zonas instrumentales y posteriormente mezclarlo siguiendo los patrones marcados por el director de la orquesta, que al fin y al cabo es el que dicta el estilo de la obra interpretada, relegando a los técnicos de la grabación a lo estrictamente técnico. Una de las técnicas de grabación más comunes en este tipo de grabaciones se basa en la comentada en la sección 2.5.1, la cual se utiliza sobre el director de la orquesta para captar la señal que servirá de referencia para el resto de señales obtenidas de los micrófonos de las distintas zonas.

3.2. Programación del Software.

Mediante la programación de un software de análisis, seremos capaces de detectar los pequeños retardos entre señales multi-microfónicas, para posteriormente corregirlos.
El entorno de programación elegido ha sido MATLAB [2], debido al conocimiento, por mi parte, de éste entorno, gracias a su extendido uso durante todo el periodo de estudio en la universidad, por gran parte de las asignaturas cursadas.
MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. Así mismo, MATLAB conjuga análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo, sin necesidad de hacer uso de la programación tradicional.

MATLAB dispone también de un amplio abanico de programas de apoyo especializados, denominados toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos toolboxes cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el 'toolbox'de procesado digital de señales, que ha sido utilizada en el desarrollo del software para este proyecto.

3.2.1. Algoritmos desarrollados.

Como hemos comentado en la introducción de la segunda sección de este tercer capítulo, la utilización de la toolbox de procesado digital de señales ha sido bastante significativa y a lo largo de este apartado nombraremos y explicaremos las funciones utilizadas de dicha toolbox.
En primer lugar, me gustaría resaltar que el objeto de este proyecto es el estudio del comportamiento de las señales de audio multi-microfónicas y la relevancia de su alineamiento desde el punto de vista auditivo y espectral.
Es por ello que no he centrado tanto la atención en el desarrollo de algoritmos exclusivos de correlación de señales, sino que he utilizado el algoritmo propuesto por la "toolbox" antes mencionada, y así centrarme en el comportamiento de estas señales.

La correlación cruzada es una medida de la similitud de dos formas de onda como una función de un retardo de tiempo aplicado a una de ellas. Suponiendo que queremos hallar la similitud entre dos señales $x_1[n]$ y $x_2[n]$ de la misma longitud N, ésta puede efectuarse mediante la suma de los productos de los correspondientes pares de puntos mediante la expresión 3.1.

$$C_{12} = \sum_{n=0}^{N-1} x_1[n]x_2[n] \tag{3.1}$$

Un resultado negativo en c_{12} indica una correlación negativa, es decir un incremento en una variable se asocia con un decremento en la otra.
Esta expresión puede indicar correlación cruzada cero y sin embargo estar las dos señales totalmente correlacionadas, como puede ser el caso de dos señales en oposición de fase. Por ello, para resolver este problema, es necesario retrasar una de ellas respecto de la otra.

$$C_{12}[k] = \sum_{n=0}^{N-1} x_1[n]x_2[n+k] = C_{21}[k] = \sum_{n=0}^{N-1} x_2[n]x_1[n-k] \tag{3.2}$$

Tal y como indica la expresión 3.2, en la primera parte, la señal $x_2[n]$ se retrasa o rota a la izquierda k intervalos de muestreo, siendo la segunda parte otra alternativa equivalente, rotar $x_1[n]$ a la derecha.

Una vez explicado en qué consiste la correlación cruzada, podemos adentrarnos en el funcionamiento y utilización del algoritmo escogido, *dsp.crosscorrelator*.
Éste devuelve la secuencia de correlación cruzada de dos entradas de tiempo discreto. Estas entradas deben tener el mismo número de muestras, es decir, deben de ser de la misma longitud.

$$r_{xy}[h] = \begin{cases} \sum_{n=0}^{N-h-1} x[n+h]y^*[n] & 0 \leq h \leq N-1 \\ r *_{xy} [h] & -(N-1) \leq h \leq 0. \end{cases} \tag{3.3}$$

Como podemos observar en la exprexión 3.3, extraída de la ayuda de Matlab [2], h es el intervalo y * denota el complejo conjugado.

Primero definimos un objeto *Hxcorr* de la clase *dsp.crosscorrelator*. Este objeto será el que lleve a cabo propiamente la correlación cruzada de las dos señales de entrada. Sien-

do éstas dos vectores de longitud N y M, el resultado de dicha correlación será una secuencia de longitud $N + M - 1$. Una vez definido este objeto, se asigna la correlación calculada mediante la función *step* y el objeto *Hxcorr* en una variable local. Este proceso se efectúa tres veces, ya que para hacer mas eficiente el algoritmo de correlación cruzada de la "toolbox" de procesado digital de señales, se extraen de las señales a analizar tres fragmentos de 1 segundo de duración (lo que corresponde con 44100 muestras en caso de que la frecuencia de muestreo de dichas señales sea de 44.1 KHz, o bien 48000 muestras en caso de que dicha frecuencia de muestreo corresponda a 48 KHz), tal y como se observa en la imagen 3.1 los tres fragmentos se almacenan en las variables *handles.sample0_1*, *handles.sample0_2* y *handles.sample0_3* para la primera señal seleccionada y *handles.sample1_1*, *handles.sample1_2* y *handles.sample1_3* para la segunda señal seleccionada. El uso de las estructuras *handles* se explica más detenidamente en el apartado 3.2.2.

El proceso de extracción de estos fragmentos se lleva a cabo automáticamente al seleccionar la señal a correlacionar cuando pulsamos la opción *Mic0* y *Mic1* en nuestra interfaz, como muestra el ejemplo de la figura 3.2 para el caso de *Mic0*.

Esto se hace debido a que las señales a correlacionar son señales muy grandes, es decir, señales con muchos datos debido a las altas frecuencias de muestreo con que nos manejamos.
Si tendríamos en cuenta todos los datos de estas señales el algoritmo de correlación cruzada tardaría varios minutos en darnos una respuestas, por lo que se ha tomado la decisión de extraer estos tres fragmentos de diferentes partes de la señal, computarlos por separado y comparar los tres resultados de la correlación cruzada para dar como resultado la media de los tres tal y como podemos observar el la imagen 3.3.

De esta forma reducimos el tiempo de cómputo a unos segundos, lo que nos permite

```
%Creo el objeto Hxcorr que luego utilizaré con step para hallar la
%correlación
Hxcorr = dsp.Crosscorrelator;

%guardo en la variable local audio_cor_1 la correlación computada con step
%entre la primera muestra de cada micrófono (sample0_1 y sample1_1)
audio_cor_1 = step(Hxcorr,handles.sample0_1,handles.sample1_1);
[V I]= max(audio_cor_1);
delay_1=I - length(handles.sample0_1);

%guardo en la variable local audio_cor_2 la correlación computada con step
%entre la segunda muestra de cada micrófono (sample0_2 y sample1_2)
audio_cor_2 = step(Hxcorr,handles.sample0_2,handles.sample1_2);
[V I]= max(audio_cor_2);
delay_2=I - length(handles.sample0_2);

%guardo en la variable local audio_cor_3 la correlación computada con step
%entre la tercera muestra de cada micrófono (sample0_3 y sample1_3)
audio_cor_3 = step(Hxcorr,handles.sample0_3,handles.sample1_3);
[V I]= max(audio_cor_3);
delay_3=I - length(handles.sample0_3);
```

Figura 3.1: Utilización del objeto *Hxcorr* para hallar la correlación cruzada entre los tres fragmentos de las dos señales.

```
% --- Executes on button press in pushbutton3_Mic0.
function pushbutton3_Mic0_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3_Mic0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[file, path] = uigetfile(('*.wav'),'Microphonic sample to process');
filewithpath = strcat(path, file);
[handles.complete_sample_0, Fs, nbits] = wavread(filewithpath);

if Fs~=handles.sampleFreq||nbits~=handles.numBits||length(handles.complete_sample_0)<12*Fs

   errordlg('The sample has not the specificed format', 'ERROR');
   pause(4);
   [file, path] = uigetfile(('*.wav'),'Microphonic sample to process');
   filewithpath = strcat(path, file);
   [handles.complete_sample_0, Fs, nbits] = wavread(filewithpath);

end

set(handles.text7_Mic0_Path,'String',num2str(filewithpath));
handles.sample0_1 = wavread(filewithpath, [Fs 2*Fs]);
handles.sample0_2 = wavread(filewithpath, [5*Fs 6*Fs]);
handles.sample0_3 = wavread(filewithpath, [10*Fs 11*Fs]);
guidata(hObject, handles);
```

Figura 3.2: Función para obtener una de las señales a correlacionar, donde podemos observar como al final se extraen los tres fragmentos y se guardan en las variables globales para su posterior uso.

```
%Hallo la media de los retrasos de cada fragmento
delay_0= [delay_1 delay_2 delay_3];
delay=round (mean(delay_0));
handles.delay = delay;

guidata(hObject, handles);
```

Figura 3.3: Media de los tres resultados para hallar un resultado común.

trabajar con mayor rapidez y aún así con mucha precisión.

Una vez determinado el retraso de una señal respecto de la otra, si lo hubiera, éste puede expresarse de forma negativa o positiva. Esto es debido, como se explica al principio de este apartado, a que dicho retraso puede expresarse como un retraso propiamente dicho, o por el contrario como un adelanto de una señal respecto de la otra, todo depende de la señal que elijamos como la primera.
Esto tiene una ventaja, y es que nos dará una idea de cual de las dos señales es la que llega en primer lugar y cual la que llega después. De todas formas esto sirve como orientación al usuario, ya que a la hora de alinear estas dos señales el resultado se interpreta correctamente tal y como muestra la imagen 3.4.

```
% Alineo Mic_0 o Mic_1 dependiendo del resultado de la correlación

if delay > 0 % Si resultado es positivo, significa que Mic_0 estará
    %restrasado "delay" muestras respecto de Mic_1
    %Aplico el retraso detectado sobre Mic_1, añadiendo al principio
    %tantos ceros como muestras esté retrasada Mic_0
    Hdelay = dsp.Delay(abs(delay));
    handles.complete_sample_delayed=step(Hdelay,handles.complete_sample_1);
    guidata(hObject, handles);

elseif delay < 0 %Si resultado es negativo, significa que Mic_1 estará
    %retrasado "delay" muestras respecto de Mic_0
    %Aplico el retraso detectado sobre Mic_0, añadiendo al principio
    %tantos ceros como muestras esté retrasada Mic_1
    Hdelay = dsp.Delay(abs(delay));
    handles.complete_sample_delayed=step(Hdelay,handles.complete_sample_0);
    guidata(hObject, handles);
```

Figura 3.4: Dependiendo del resultado hallado sea positivo o negativo, se interpreta de una manera u otra para alinear correctamente ambas señales

Como vemos, si el resultado el positivo, se interpreta que la primera señal (cargada en *Mic0*) estará retrasada el número de muestras hallado y almacenado en *delay* respecto de la segunda señal (cargada en *Mic1*).

Una vez interpretado el resultado positivo, se procede a linear ambas señales. Aquí hacemos uso de la "toolbox" de nuevo, utilizando la clase *dsp.Delay*. Este bloque de retardo, tal y como explica la documentación de Matlab [2], retrasa una entrada discreta en el tiempo por el número de muestras especificados en las unidades y los parámetros de retardo. El valor de retardo debe ser un valor entero mayor o igual que cero.

De la misma forma que con *dsp.crosscorrelator*, creo un objeto *Hdelay* de la clase *dsp.Delay* al cual le paso el parámetro *delay* (obtenido mediante la correlación cruzada) en valor absoluto (*abs (delay)*), lo que generará el retraso necesario que luego se incorporará al comienzo de la señal *Mic1* (para este primer caso en el que el retraso hallado es positivo). Esta señal retrasada se almacena en *handles.complete_sample_delayed* para su posterior reproducción y guardado.
Se ha optado por retrasar la señal adelantada (en este caso *Mic1*) ya que de otra forma tendríamos que adelantar la señal retrasada (en este caso *Mic0*) quitando muestras a la señal *Mic1*. Se prefiere que no halla ninguna pérdida de información respecto a las señales, por lo que se elige añadir retardos a sustraer muestras.

Para el segundo caso, si el resultado es negativo, se interpreta que la segunda señal (cargada en *Mic1*) estará retrasada el número de muestras hallado y almacenado en *delay* respecto de la primera señal (cargada en *Mic0*).
A partir de aquí el proceder es igual que para el caso anterior pero a la inversa, es decir, se aplica el retraso *Hdelay* en valor absoluto, al comienzo de la señal *Mic0*, para alinear ambas señales.

3.2.2. Interfaz de usuario.

Para desarrollar la Interfaz también se ha utilizado MATLAB, ya que posee una herramienta sencilla de diseño, denominada GUIDE.
GUIDE es un entorno de programación visual para realizar y ejecutar programas que

necesiten ingreso continuo de datos. Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++. Una aplicación GUIDE consta de

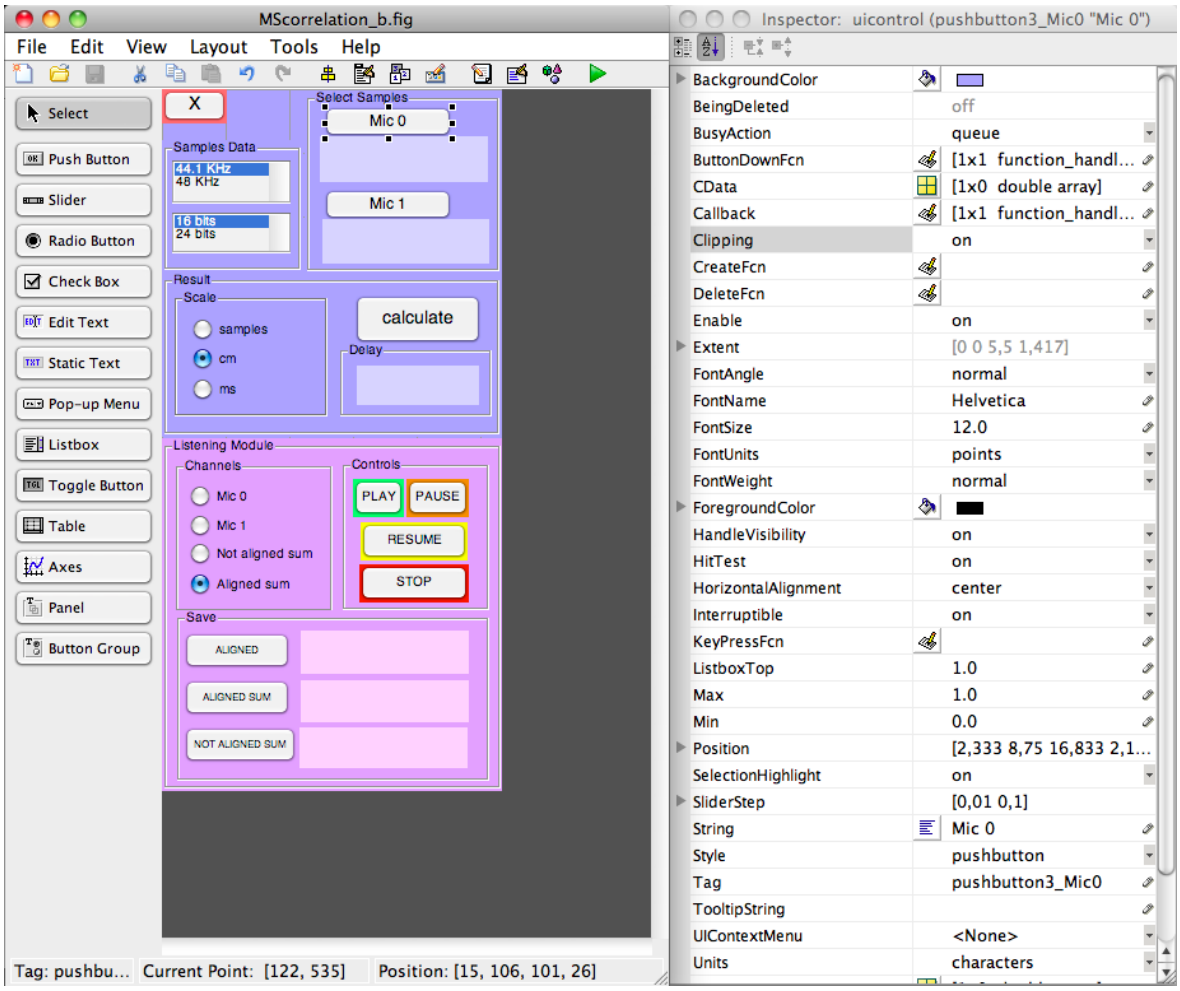


Figura 3.5: Entorno de programación visual GUIDE. A la izquierda se observa el área de diseño y a la derecha aparece el inspector de propiedades, único para cada elemento

dos archivos: .m y .fig. El archivo .m contiene el código con las órdenes de los botones de control de la interfaz y el archivo.fig, como muestra la Figura 3.5, contiene los elementos gráficos. Cada vez que se añada un nuevo elemento en la interfaz gráfica, se genera su correspondiente código automáticamente en el archivo .m.

El manejo de datos entre el archivo.fig (elementos gráficos de la aplicación) y el archivo.m se lleva a cabo mediante el uso de una estructura. Todos los valores de las propiedades de los elementos (color, valor, posición, string...) y los valores de las variables transitorias del programa se almacenan en dicha estructura. Todos estos valores se acceden mediante un identificador para todos tal y como muestra el código de la figura 3.6, éste se asigna en:

$$handles.output = hObject;$$

handles, es nuestro identificador a los datos de la aplicación. La definición del identi-

```
% --- Executes when selected object is changed in uipanel1_scale.
function uipanel1_scale_SelectionChangeFcn(hObject, eventdata, handles)
%   hObject:  handle to the selected object in uipanel1_scale
```

Figura 3.6: Ejemplo de estructura *handles* usada en panel de selección de escala para acceder a los datos almacenados en estas estructuras.

ficador se salva como se muestra en el código de la imagen 3.7 con la instrucción:

$$guidata(hObject, handles);$$

siendo *guidata* la sentencia para salvar los datos de la aplicación y la cual debemos añadir después de cada modificación de los datos de las variables de la estructura, para garantizar que cualquier cambio o asignación queda almacenado. Asimismo para llevar a cabo la asignación y/o obtención de los valores de los elementos de la aplicación se utilizan las sentencias *set* y *get*, como podemos observar en el código de la figura 3.7.

```
set(handles.text8_Mic1_Path,'String',num2str(filewithpath));
handles.sample1_1 = wavread(filewithpath, [Fs 2*Fs]);
handles.sample1_2 = wavread(filewithpath, [5*Fs 6*Fs]);
handles.sample1_3 = wavread(filewithpath, [10*Fs 11*Fs]);
guidata(hObject, handles);
```

Figura 3.7: Ejemplo de sentencia *set* usada para mostrar la ruta de obtención de una de las muestras de audio.

3.3. Funcionamiento y manejo del software

Una vez explicados los conceptos anteriores, para manejar y almacenar datos en las estructuras *handles* y cómo acceder a ellos, podemos pasar a explicar los diferentes módulos en los que se divide el software desarrollado. Éste se divide en cuatro módulos principales bien diferenciados.

Módulo 1: Selección de señales de audio

En primer lugar se encuentra el módulo (mostrado en la figura 3.8)de carga de señales a nuestro programa, mediante el cual elegiremos de entre todos nuestros archivos, los que deseemos correlacionar.

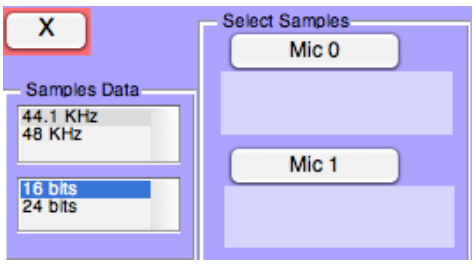


Figura 3.8: Módulo de selección de señales para correlacionar.

Dentro de este módulo se encuentran la selección de los parámetros de las señales seleccionadas, tanto *Frecuencia de muestreo*, seleccionable entre 44.1 KHz y 48 KHz; y *Número de bits*, seleccionable entre 16 y 24. Actualmente todas las señales de audio

manejadas en el programa son de 44.1 KHz y 16 bits, debido al menor numero de datos de estas señales respecto a las señales de 48 KHz y 24 bits, lo que repercute en el tiempo de cómputo del algorítmo de correlación cruzada explicado en el apartado 3.2.1.

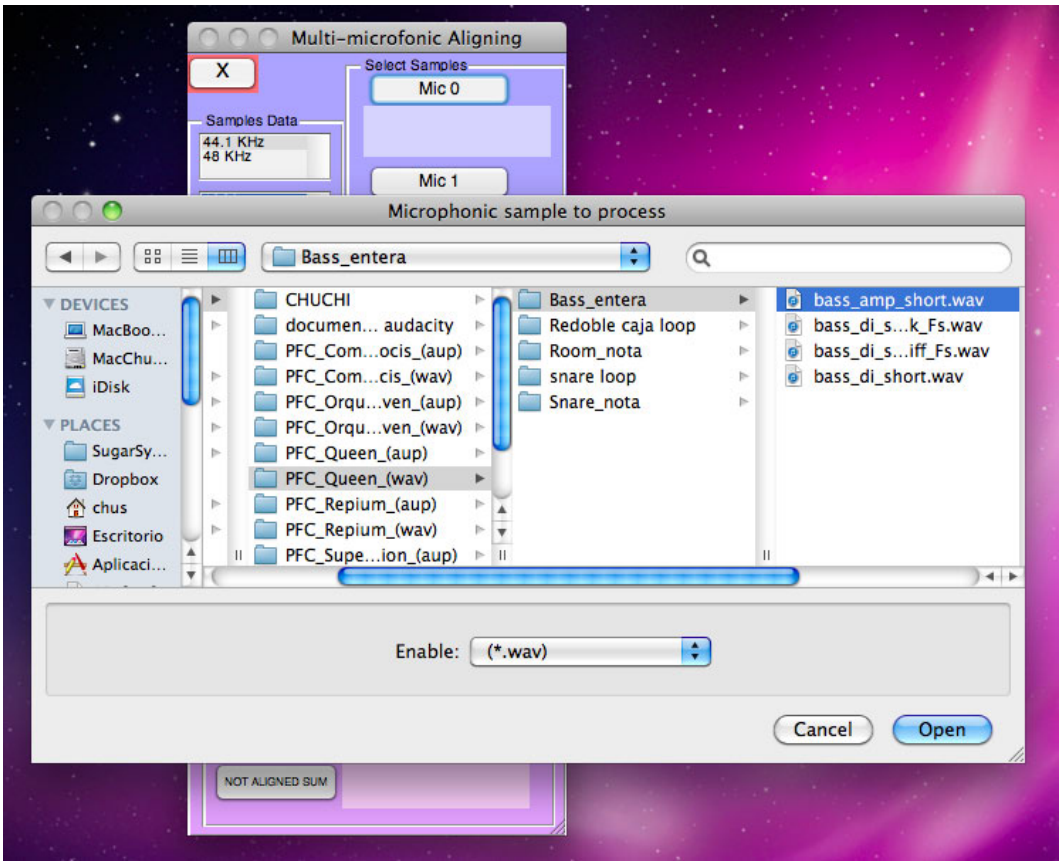


Figura 3.9: Ventana de elección de señal de audio.

A la derecha de la selección de los parámetros, se muestran dos botones *Mic0* y *Mic1* los cuales, al pulsarlos nos aparece una ventana desde la cual podemos elegir las señales de audio que deseemos correlacionar en nuestro programa (imagen 3.9), y una vez seleccionada se muestra la ruta de acceso a dicha señal de audio en un cuadro de texto bajo sendos botones. De esta forma sabremos en todo momento que dos señales de audio estamos correlacionando, como se puede observar en la imagen 3.10.

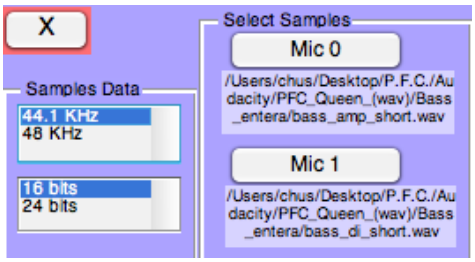


Figura 3.10: Ruta de acceso de ambas señales mostrada en los cuadros de texto.

En caso de que las señales elegidas no se ajusten a los parámetros definidos en la parte izquierda de este módulo, aparecerá un cuadro de diálogo indicando precisamente este error en la elección de la señal de audio (imagen 3.11), e inmediatamente después aparecerá de nuevo la ventana desde la cual volveremos a elegir otra señal que se ajuste a los parámetros antes establecidos, y así poder continuar con la correlación entre ambas. Por último, destacar el botón *X* situado en la parte superior izquierda, que sirve para salir del programa.

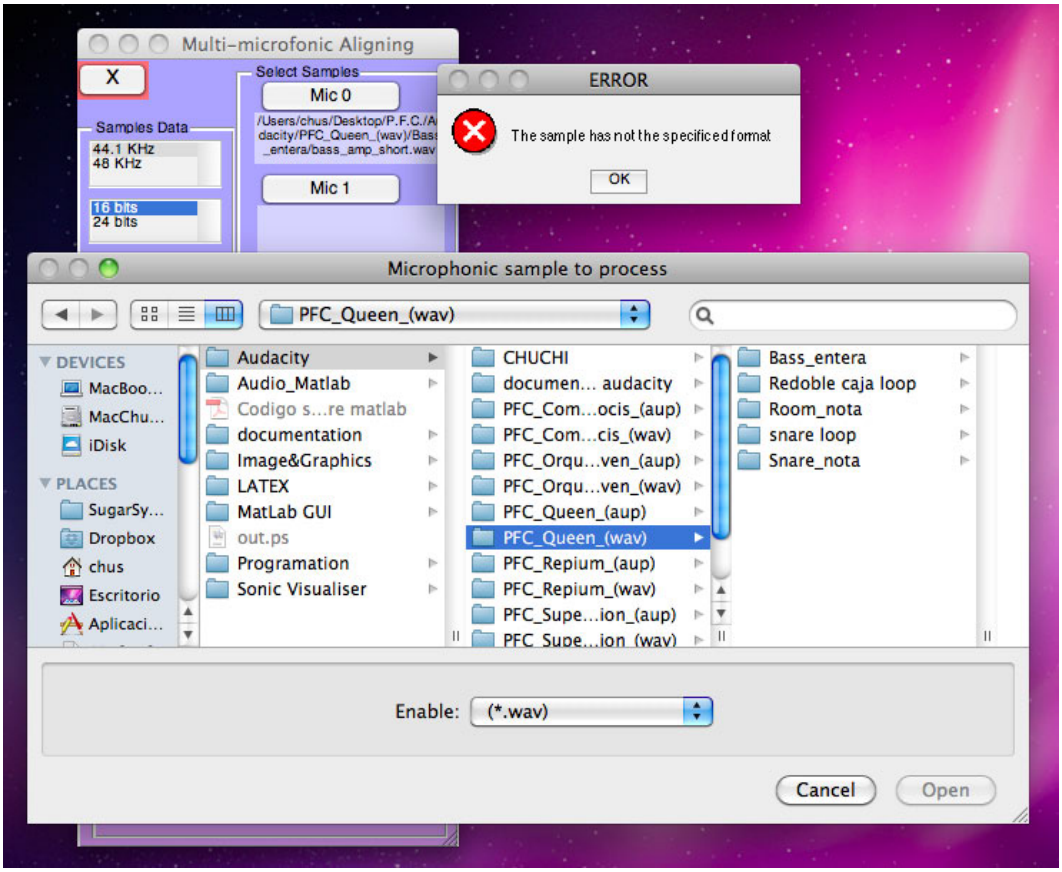


Figura 3.11: Cuadro de diálogo indicativo del error en las señales de audio seleccionadas.

Módulo 2: Resultados

En segundo lugar, se sitúa el módulo de cálculo de resultados, mostrado en la magen 3.12.

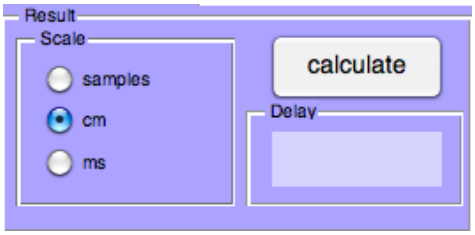


Figura 3.12: Módulo de cálculo de resultados.

Dentro de este módulo se encuentra en la parte izquierda un submenú de escala con tres opciones a elegir: *samples* (muestras), *cm* (centímetros) y *ms* (milisegundos). Estas tres opciones hacen referencia al resultado de la correlación cruzada obtenido al pulsar la tecla *calculate* en la parte derecha del módulo de resultados. Si elegimos la opción *samples* nuestro resultado expresará el número de muestras que una señal este retrasada respecto de la otra (imagen 3.13), si marcamos la opción *cm* nuestro resultado expresará el número de centímetros que estarían separados los micrófonos que recogieron las señales de audio correlacionadas (imagen 3.14), y si por último marcaríamos la opción *ms* nuestro resultado nos indicaría el tiempo de retardo existente entre ambas muestras de audio expresado en milésimas de segundo (imagen 3.15).

Este segundo módulo esconde una particularidad, y es que no existe ningún botón que permita alinear ambas señales, sino que se ha optado por realizar automáticamente este proceso cuando seleccionamos el botón *calculate* y se almacena en una variable global para su posterior uso dentro del programa.
Al pulsar la tecla *calculate*, mencionada anteriormente, para calcular la correlación entre ambas señales de audio, aparece un cuadro de diálogo que indica que la correlación

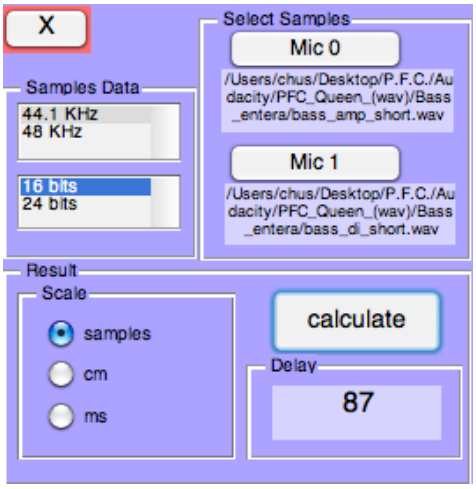


Figura 3.13: Cálculo del retardo en muestras.

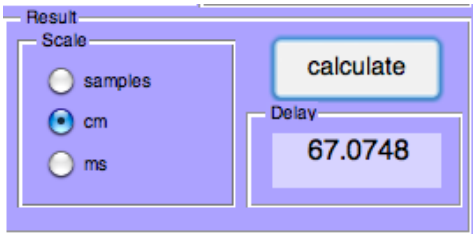


Figura 3.14: Cálculo del retardo en centímetros

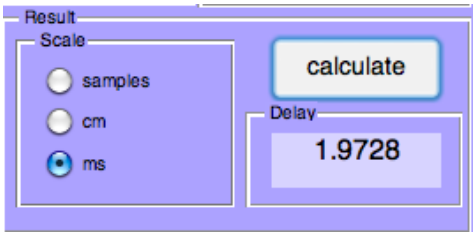


Figura 3.15: Cálculo del retardo en milésimas de segundo.

está en progreso. Este proceso tardará unos segundos, debido a la cantidad de datos que hay que computar, y se ha pensado en poner este aviso para dar a entender al usuario que el proceso se está llevando a cabo con normalidad por el programa.

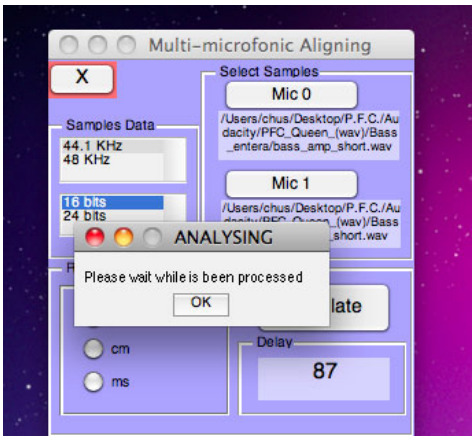


Figura 3.16: Cuadro de diálogo indicativo del proceso de correlación llevado a cabo por el programa.

Módulo 3: Escucha previa

En tercer lugar tenemos el módulo de escucha previa de las señales de audio. Aquí disponemos en la parte derecha de un cuadro de controles, mediante los cuales controlamos las señales de audio seleccionadas en el submenú de la parte izquierda, tal y como nos muestra la imagen 3.17.

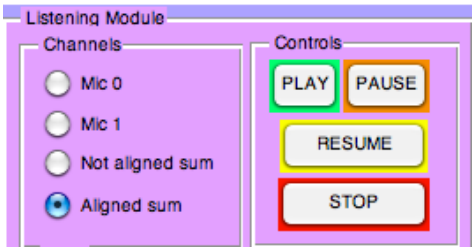


Figura 3.17: Módulo de reproducción de las señales de audio.

Dentro del cuadro de controles podemos encontrar el botón *PLAY*, el cual sirve para

reproducir la señal, a su lado está el botón *PAUSE*, con el que podemos pausar la reproducción, debajo de ambos se encuentra el botón *RESUME*, que permite reanudar la reproducción desde el punto pausado, y por último debajo de todos los demás está *STOP*, que para la reproducción por completo.

En el submenú de la parte izquierda encontramos cuatro opciones, que nos permiten escoger entre las distintas señales de audio para reproducirlas. Podemos seleccionar *Mic 0* para escuchar la primera señal de audio seleccionada en el módulo 1, *Mic 1* para reproducir la segunda seña de audio seleccionada en el módulo 1, *Not aligned sum* (suma no alineada) para escuchar la suma de das los señales de audio anteriores sin alinear, es decir, tal y como se cargan en el módulo 1, y por último *Aligned sum* (suma alineada), para escuchar la suma alineada de las señales anteriores con respecto al retraso hallado en el módulo 2.

Este tercer módulo tambien esconde una particularidad, y es que se ha optado por realizar automáticamente proceso de suma de señales cuando seleccionamos tanto la opción *Not aligned sum* como *Aligned sum*, como podemos ver en el código de la imagen 3.18.

```
case 'radiobutton16_Aligned'
    % Code for when radiobutton2_cm is selected.
    handles.Mic0=0;
    handles.Mic1=0;
    handles.NotAlign=0;
    handles.Align=1;
    if handles.delay > 0 % si el retraso es positivo, se sumará la señal
                         % retrasada con la señal seleccionada en Mic0,
        handles.complete_sum_align=handles.complete_sample_0+handles.complete_sample_delayed;
        handles.complete_sum_audioplayer_align=audioplayer(handles.complete_sum_align,handles.sampleFreq);
        guidata(hObject, handles);

    elseif handles.delay < 0 % si el retraso es negativo, se sumará la señal
                             % retrasada con la señal seleccionada en Mic1,
        handles.complete_sum_align = handles.complete_sample_1 + handles.complete_sample_delayed;
        handles.complete_sum_audioplayer_align=audioplayer(handles.complete_sum_align,handles.sampleFreq);
        guidata(hObject, handles);

    end
    guidata(hObject, handles);
```

Figura 3.18: Código asignado a la selección *Aligned sum* del submenú.

Módulo 4: Guardado de señales

Por último tenemos el módulo de salvado de señales. En este módulo tenemos tres opciones de guardado, tal y como muestra la imagen 3.19. El primero de los tres bo-

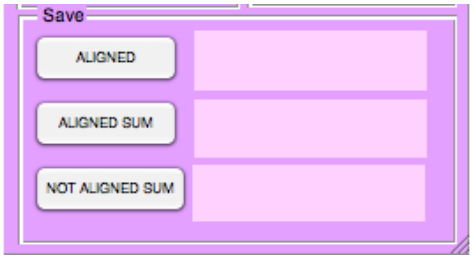


Figura 3.19: Módulo de guardado de señales.

tones, *ALIGNED*, permite guardar en el directorio que escojamos, como muestra la imagen 3.20, la señal de audio alineada, es decir, la señal de audio a la que hemos aplicado el retraso calculado en el segundo módulo.
En la ventana que se abre al pulsar el botón *ALIGNED*, se puede observar que lleva como título *Save Mic 1 aligned sample as* (guardar muestra alineada de Mic 1 como) y en la casilla *Save As* donde se escribe el nombre con el que se quiere guardar el archivo, el programa te sugiere el nombre *Mic_1_delayed*.

Esto se ha hecho así para que al guardar la señal alineada se sepa cual de las dos señales se ha alineado con respecto a la otra. Tal es así, que en este caso en particular se ha alineado la señal elegida en Mic 1, por lo que si queremos guardarla con el nombre que tenía el archivo de audio cuando lo cargamos al programa, tan sólo tenemos que mirar en el cuadro de texto bajo el botón *Mic1* en el módulo 1, ver en la parte final el nombre del archivo, y en mi caso guardarlo con el mismo nombre seguido de algún identificativo, tal y como se muestra en la imagen 3.21.

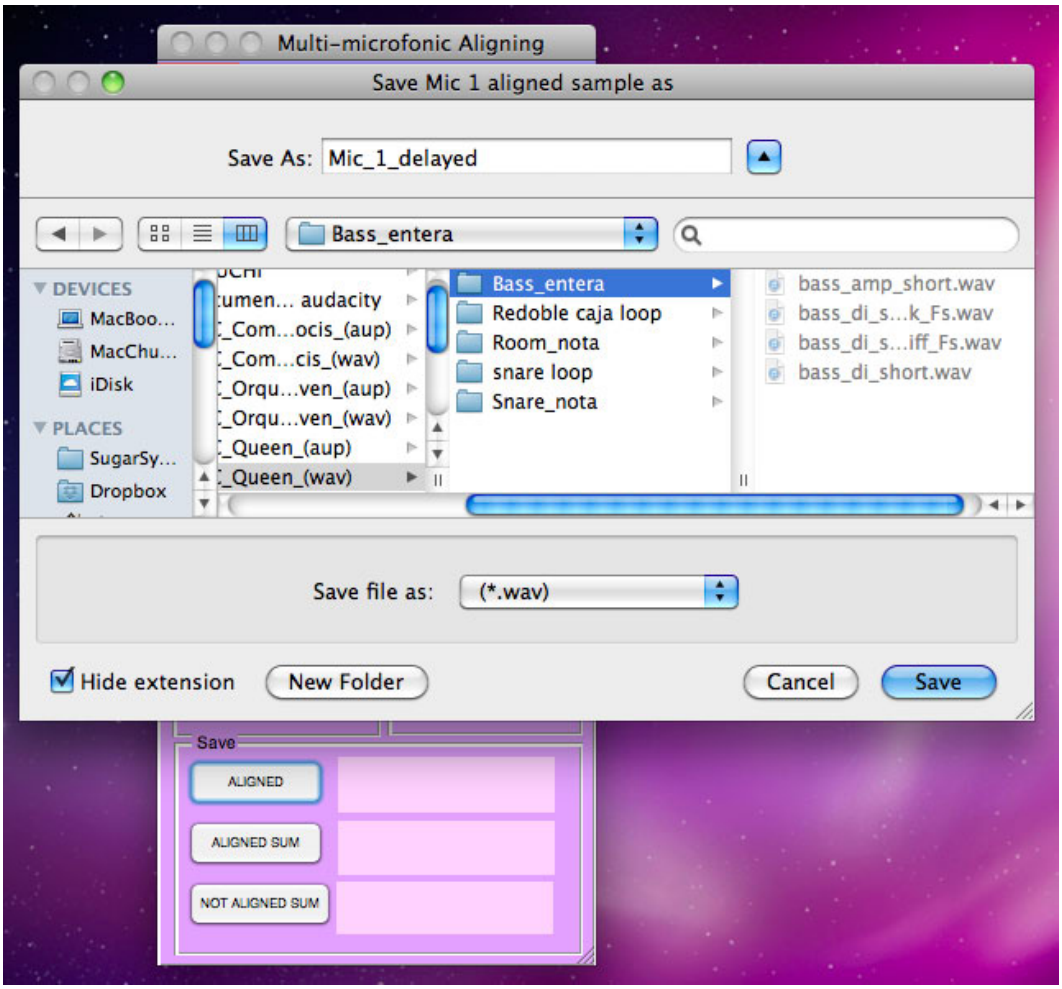


Figura 3.20: Ventana emergente donde almaceno el archivo de audio en el directorio escogido.

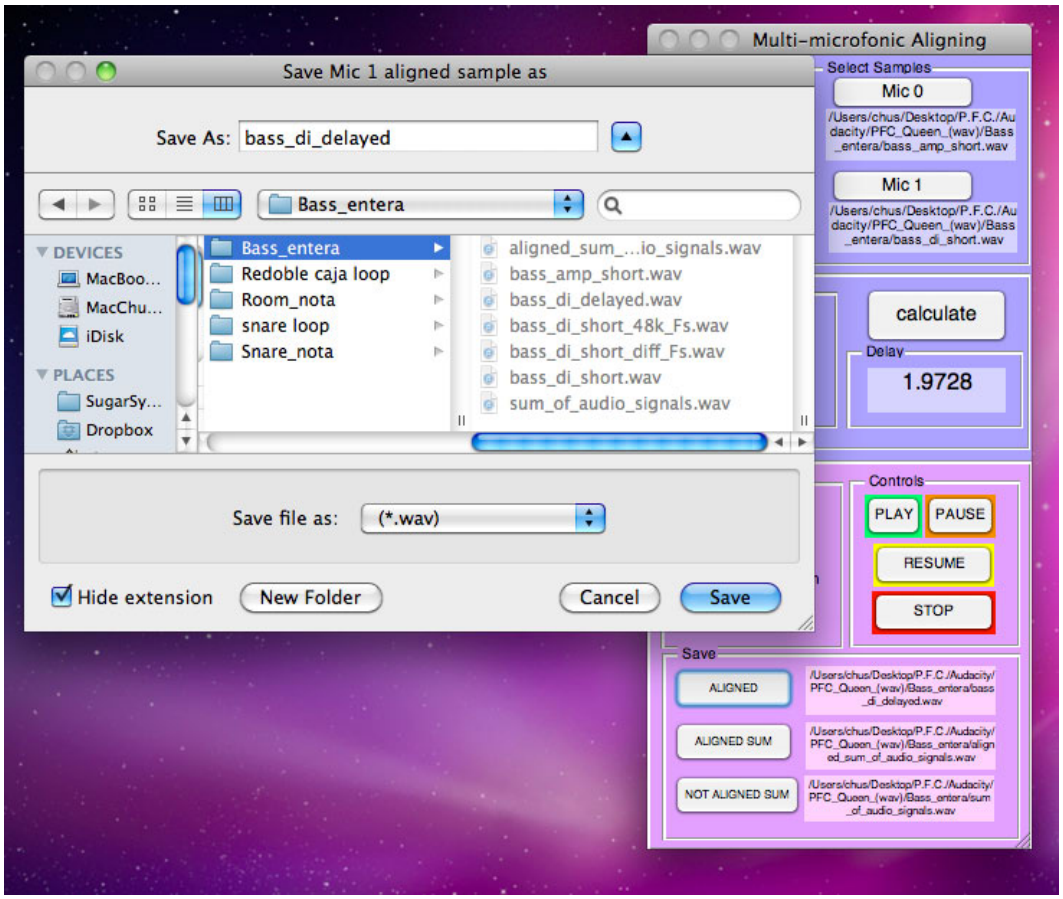


Figura 3.21: Código asignado a la selección *Aligned sum*.

El segundo de los tres botones, *ALIGNED SUM*, permite guardar en el directorio que escojamos, la señal de audio resultante de la suma de la señal alineada con la otra señal de referencia, es decir, de las dos señales correlacionadas tras alinearlas.
Por último, el tercer botón, *NOT ALIGNED SUM*, permite guardar la señal de audio resultante de la suma de ambas señales correlacionadas sin alinear.
Del mismo modo que en el caso del Módulo 1, al lado de los botones se encuentras unos cuadros de texto donde aparecerá la ruta de acceso a los directorios donde se han almacenado los archivos de audio (imagen 3.22).

Figura 3.22: Guardado de las tres señales en sus directorios.

Se han incluído estas dos opciones en la interfaz del programa, debido a la dificultad que resulta de mostrar las diferencias auditivas entre señales de audio en un medio escrito, como es este que tiene ante su mirada. Por ello se ha incluído esta forma de salvar las señales y así poder analizarlas mediante un analizador de espectrograma y mostrar visualmente las diferencias auditivas detectadas entre la suma alineada y la suma no alineada de señales. Sobre ésto se hará más hincapié en el siguiente capítulo, en el que pondremos en práctica el software desarrollado en diferentes situaciones, explicando las diferencias apreciadas.

Por último mostrar en la imagen 3.23 su funcionamiento global, después de su explicación paso a paso de cada uno de sus módulos, este sería el aspecto final, tras la selección

de señales, el cálculo de su correlación, la escucha previa y el salvado de las señales necesarias.

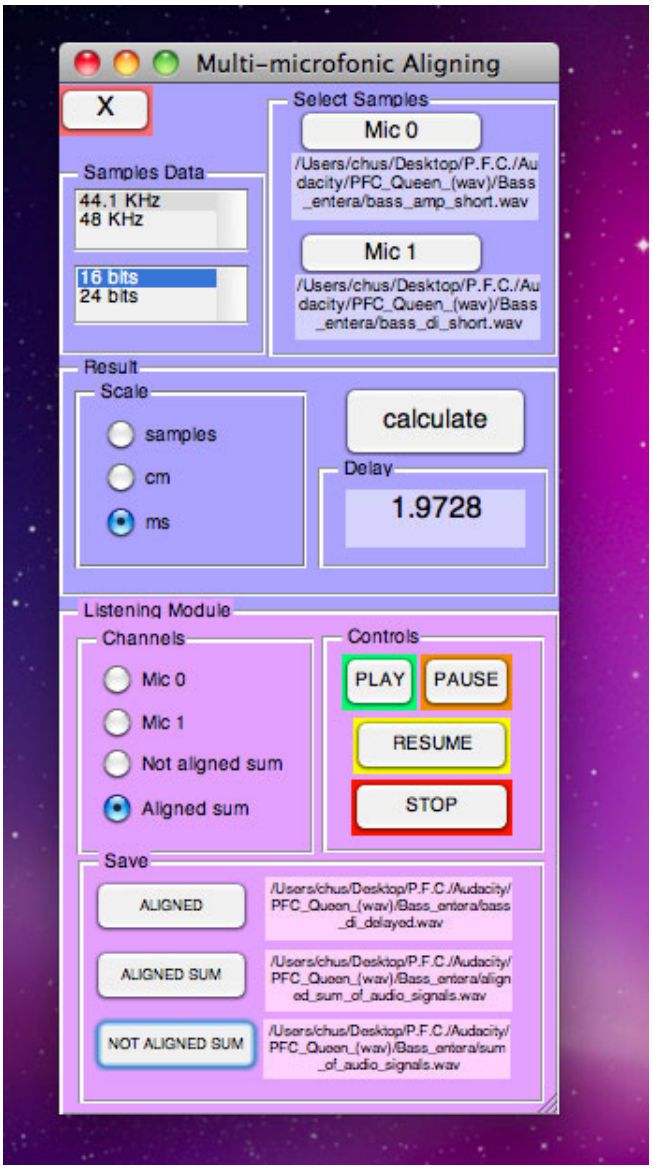


Figura 3.23: Visualización del funcionamiento global del programa.

Capítulo 4

Resultados experimentales

En este capítulo se van a presentar los resultados de la aplicación del software diseñado, y explicado en el capítulo anterior, en diversos escenarios tales como:

- Señales digitals internas
- Grabaciones dentro del estudio.
- Grabaciones fuera del estudio.

Para mostrar las diversas diferencias entre las señales de audio obtenidas tras utilizar nuestro software, utilizaremos una herramienta muy útil en este tipo de casos, usaremos espectrogramas para mostrar las diferencias auditivas en el espectro de frecuencias audibles. ya que de otro modo sería muy complicado a través de un medio escrito como es este proyecto.
Para obtener los espectrogramas de las señales de audio obtenidas con el software se ha utilizado un software llamado *Sonic Visualiser*, un software diseñado por *Chris Cannan y la Universidad Queen Mary de Londres*[3]. Se ha escogido este software por su sencillez, además de estar desarrollado por compañeros de Londres, y por que se encuentra bajo la licencia *creative commons*, lo que nos permite usarlo con todo su consentimiento.

El el uso de este software cabe destacar que los parámetros elegidos para la correcta visualización de los espectrogramas son: un enventanado tipo Hanning con una ventana de 2048 muestras y un solapamiento del 87,5 %, una escala referenciada en dBv, adjuntando una escala de colores junto al espectrograma expresada en dBFS (deciBelios de fondo de escala), como referencia para interpretar los colores a lo largo del espectrograma. Ésta escala se muestra de forma linear para el primer apartado de esta sección (señales digitales internas) ya que se hace más visible la representación de los fenómenos estudiados con el ruido blanco de forma linear, al estar éste repartido de manera uniforme en todo el espectro de frecuencias. En los sucesivos dos apartados, como las señales a visualizar con los espectrogramas son señales de audio, se ha prefe-

rido mostrar en una escala logarítmica, ya que se ajusta más visualmente a la realidad auditiva, dando más peso a las frecuencias graves que a las frecuencias agudas.

4.1. Señales digitales internas

Para llevar a cabo los resultados de esta sección se ha tomado como muestra de audio un fragmento de ruido blanco de aproximadamente unos 15 segundos de duración. Para ello se a utilizado el programa de edición de audio multi-track *Audacity*, un programa de software libre, desarrollado por un grupo de voluntarios y distribuido bajo la Licencia Pública General de GNU (GPL)[4].

Lo que se hecho con *Audacity* es generar una señal de ruido blanco y luego retrasarla un número indeterminado de muestras creando 5 nuevas señales (como muestra la imagen 4.1) retrasadas un número diferente de muestras respecto de la primera señal de ruido blanco, que es la que tomaremos como referencia.

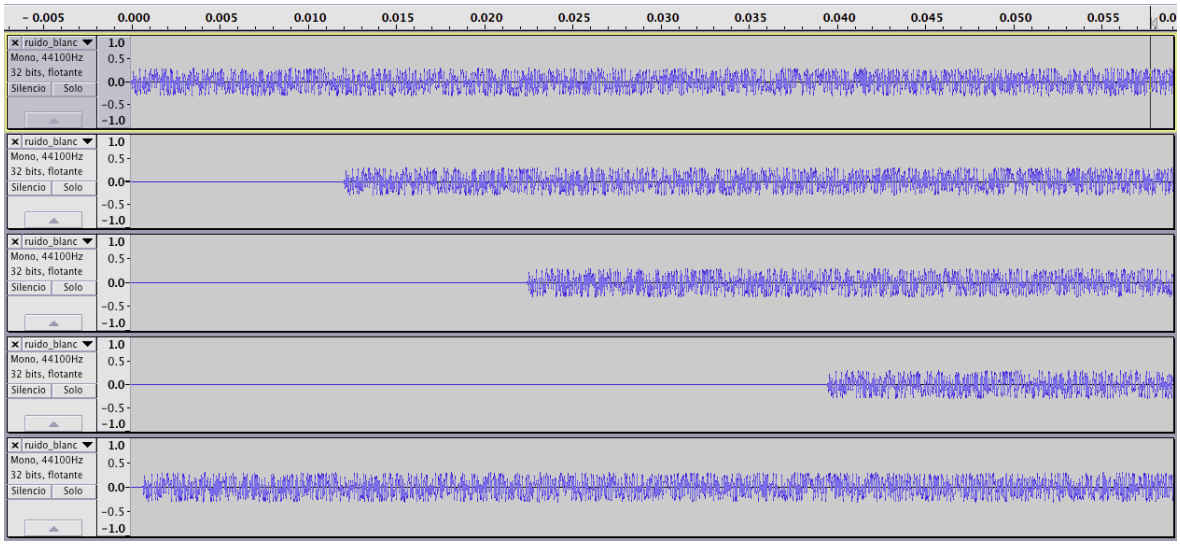


Figura 4.1: Señales de ruido blanco generadas con *Audacity*.

De este modo, demostraremos la existencia del filtro peine o más conocido como *comb filter* en inglés, explicado en el capítulo de conceptos teóricos. Para ello analizaremos, con el software diseñado, las señales de ruido blanco, tomando como referencia la primera señal de ruido blanco, a partir de la cual retrasamos las demás, teniendo así una medida objetiva de los distintos desfases y cancelaciones de frecuencias.

En primer lugar, analizamos la señal de ruido blanco de referencia con ella misma, para comprobar el resultado de nuestro software, que tal y como muestra la imagen 4.2 el resultado de la correlación da 0 muestras, lo que quiere decir que no existe retraso entre ellas puesto que son la misma señal. Una vez hecho esto, pasamos a analizar las demás señales de ruido blanco, de las cuales extraemos los espectrogramas de las más destacables, para observar mejor las diferencias entre la suma normal de las dos señales correlacionadas y la suma alineada de éstas.
La más apreciable resulta, la que menos muestras está retrasada, en este caso correlacionamos las señales de ruido blanco de referencia con la que ocupa el quinto lugar según la figura 4.1, dando como resultado del retraso (expresado en muestras) el mostrado en la imagen 4.3.

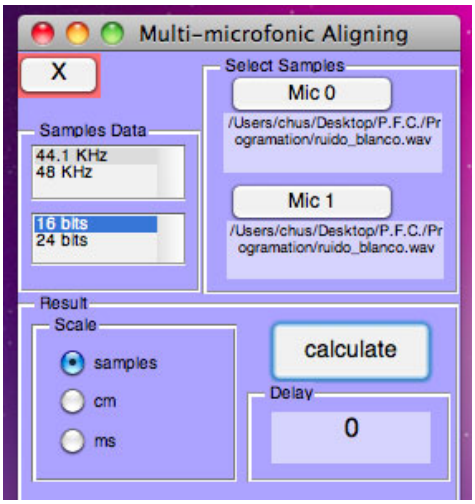


Figura 4.2: Señal de ruido blanco de referencia correlacionada con ella misma.

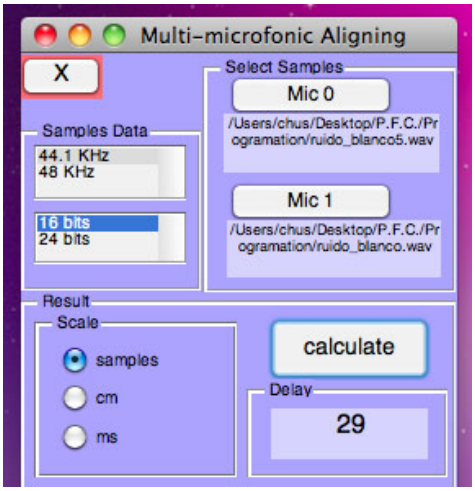


Figura 4.3: Retraso entre señales de ruido blanco de referencia y número cinco.

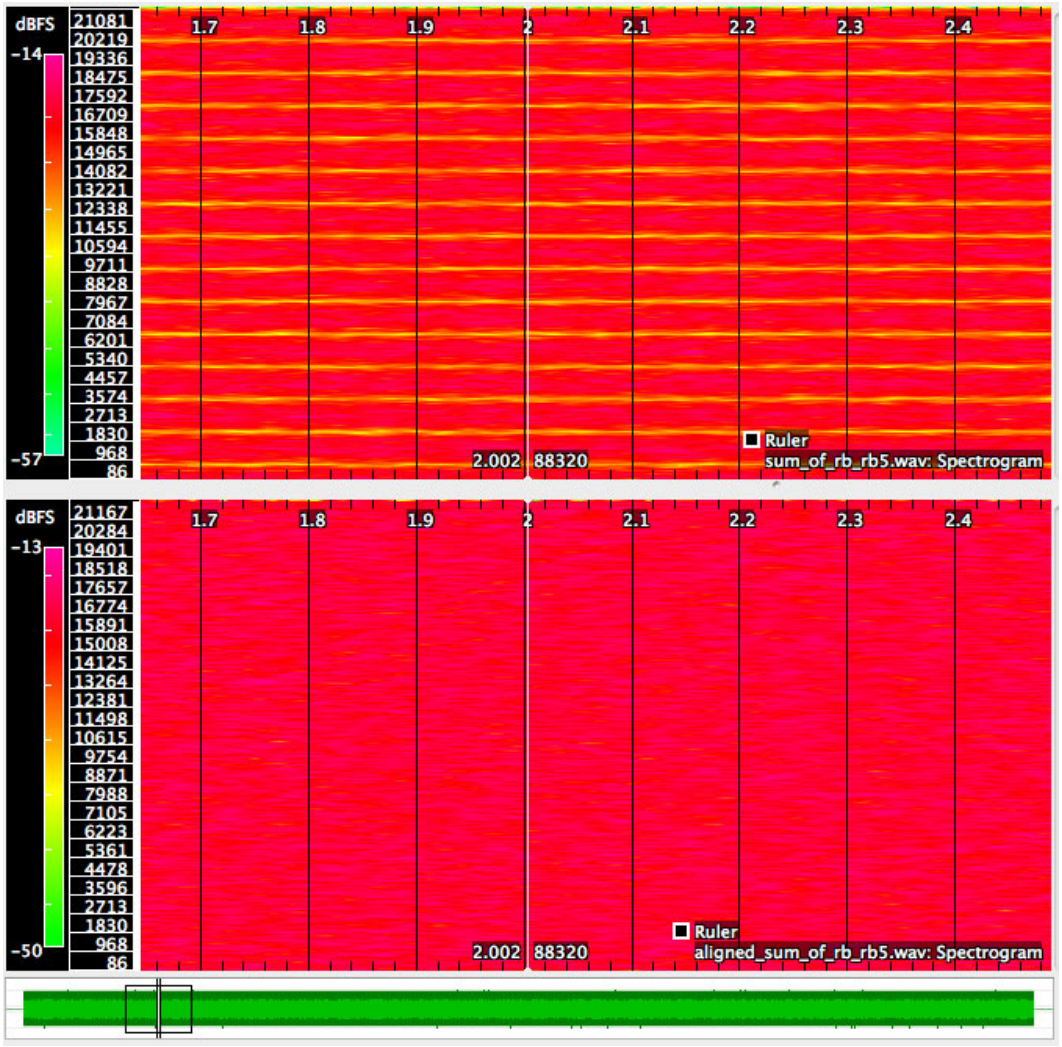


Figura 4.4: Señal de ruido blanco de referencia correlacionada con la señal de ruido blanco en quinto lugar, según la figura 4.1.

Con las señales ya correlacionadas, sabiendo el retraso de una respecto de la otra, podemos escuchar las diferencias existentes entre la suma alineada y sin alinear dentro del software desarrollado, pero para poder observar las diferencias visualmente, guardamos las señales para posteriormente analizarlas con el software mencionado anteriormente *Sonic Visualiser*, obteniendo los resultados mostrados en la imagen 4.4.

El espectro de arriba corresponde a la suma no alineada de las dos señales de ruido

blanco. En él se observan una líneas de color amarillo que corresponden a una caída del nivel en deciBelios, a ciertas frecuencias, que son las que se anulan o se perciben muy levemente. El espectro de abajo corresponde a la suma alineada de ambas señales, y como se puede observar, ya no aparecen esas líneas de color amarillo, pues no hay ningún valle de frecuencia, es decir, al corregir el desfase, ya no desaparece ninguna frecuencia, por lo que aparecen todos los niveles aparecen bien repartidos por todo el espectro.

De igual forma que la anterior, correlacionamos ahora la señal de ruido blanco de referencia con la señal de ruido blanco en segundo lugar, según la figura 4.1, dando como resultado del retraso expresado en muestras el mostrado en la imagen 4.5.

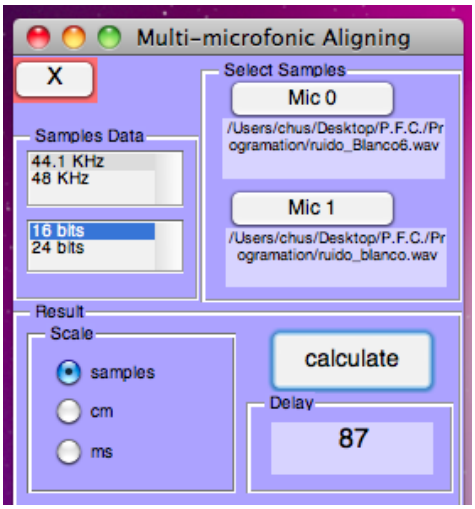


Figura 4.5: Retraso entre señales de ruido blanco de referencia y número dos.

A través de espectrograma (imagen 4.6 observamos unas líneas de color rosa, que igual que en el caso anterior, corresponden con las caídas de nivel en ciertas frecuencias, que son las que conforman el filtro peine. Como se puede comprobar en el espectro de abajo, al alinear las señales, desaparecen esas líneas, debido a que desaparecen las cancelaciones de frecuencias.

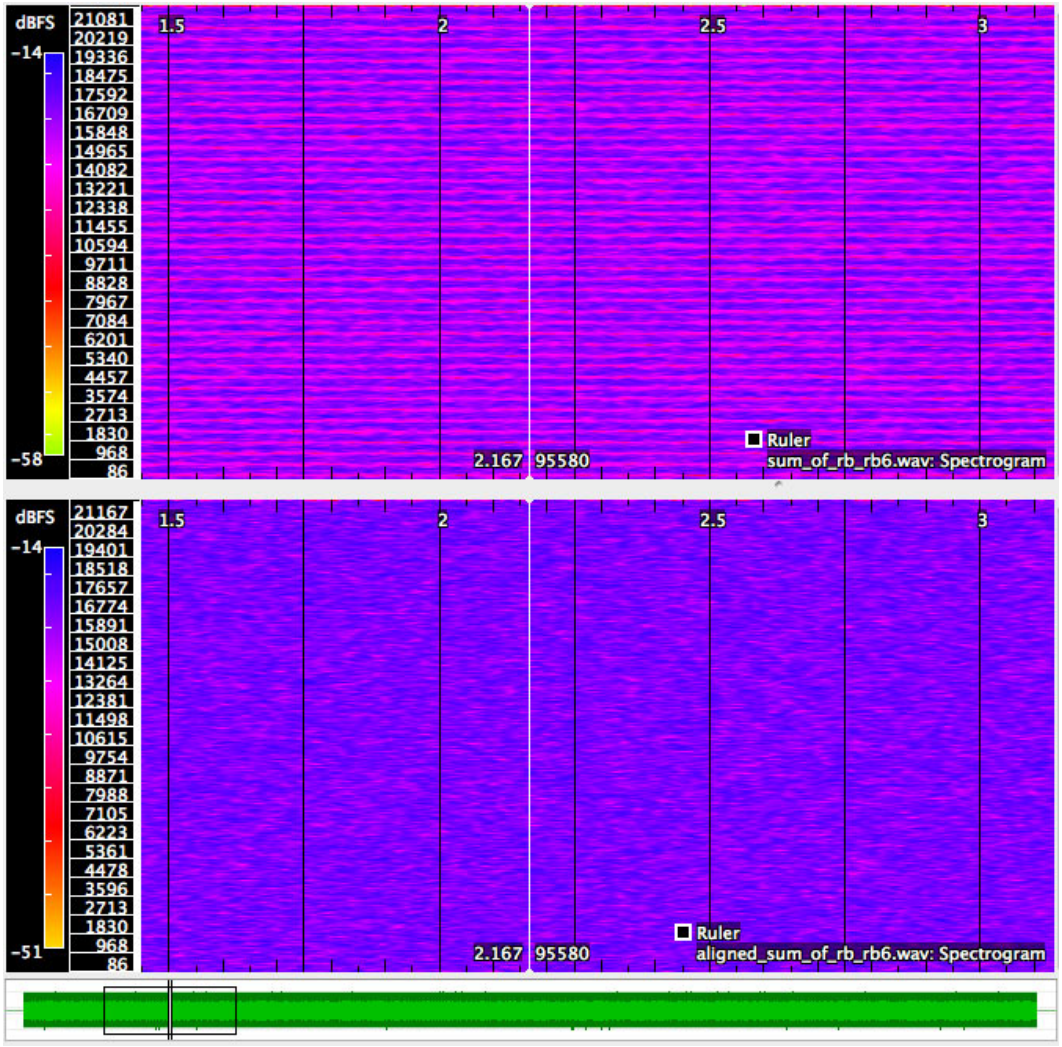


Figura 4.6: Señal de ruido blanco de referencia correlacionada con la señal de ruido blanco en segundo lugar, según la figura 4.1.

Por último, el resultado del retraso entre las señales de ruido blanco de referencia y la que ocupa el tercer lugar en la figura 4.1 se muestra en la imagen 4.7, y es en la que menos apreciable resulta el filtro peine, ya que debido al carácter de ruido blanco y el retraso tan grande, no ocurre el fenómeno observado anteriormente. Sin embargo si se produce un efecto audible.

Como se observa en el espectrograma de las señales resultantes de la correlación con

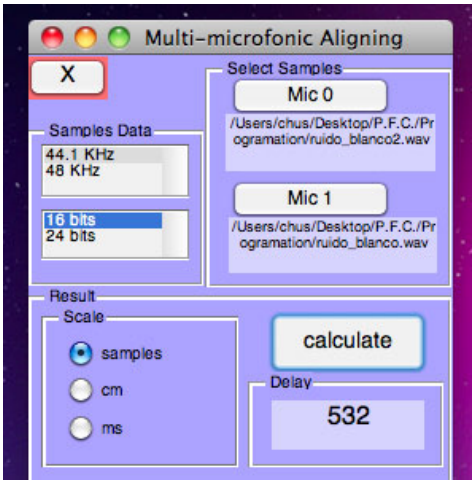


Figura 4.7: Retraso entre señales de ruido blanco de referencia y número tres.

nuestro software, mostrado en la imagen 4.8, no se aprecia ninguna caída de nivel específica a una cierta frecuencia, sino que se aprecian una caídas de nivel por zonas de frecuencias.

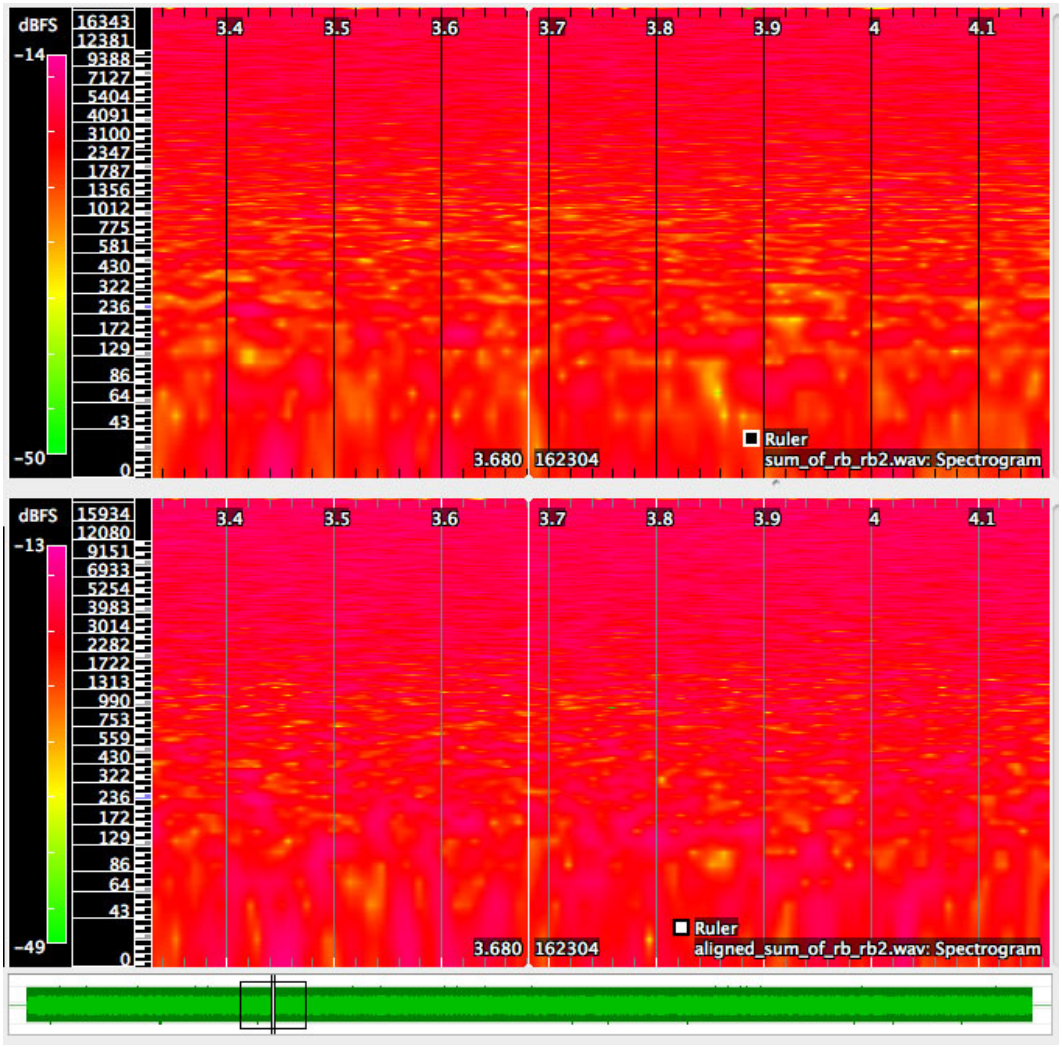


Figura 4.8: Señal de ruido blanco de referencia correlacionada con la señal de ruido blanco en tercer lugar, según la figura 4.1.

4.2. Grabaciones dentro del estudio

Para esta sección he utilizado varias grabaciones de estudio, algunas de ellas cedidas por Mario Toca, compañero y amigo al que agradezco haberme prestado su tiempo y dejarme usar para este proyecto algunas de las pistas de la grabación del grupo *Repium* llevado a cabo en su estudio, y otras cedidas Helios Vega, durante muchos años jefe y ahora compañero y amigo, al que agradezco la motivación de seguir en esto del sonido.

4.2.1. Alineamiento en las tomas de Batería

Para las tomas de batería se ha usado una de las técnicas microfónicas explicadas en el capítulo de conceptos teóricos. Como muestra la figura 4.9, se ha colocado un Par

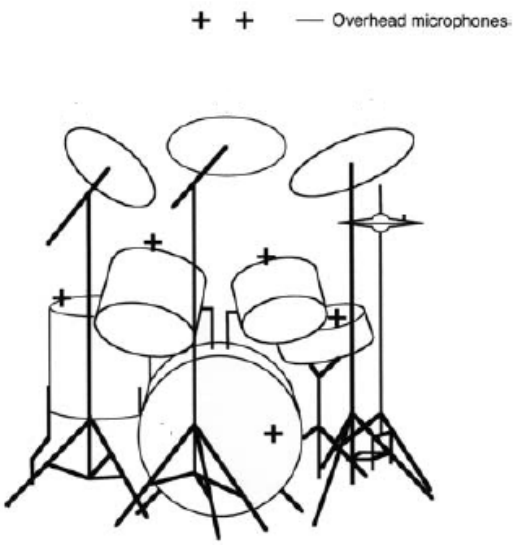


Figura 4.9: Técnica seguida para la grabación de las pistas de batería

estero XY sobre la cabeza del percusionista, lo que en inglés se denomina *Over Head (OH)* haciendo así referencia al derecho como *OHR* y al izquierdo *OHL*. Además se ha

colocado un micrófono para recoger la caja (en inglés *snare*, y otros tres para recoger cada uno de los timbales o *toms*, según su denominación en inglés. Por último se ha colocado in micrófono dentro del bombo o en inglés *kick*. Todo esto sigue una estructura típica de grabación de sets de batería, tal y como muestra la figura 4.10.

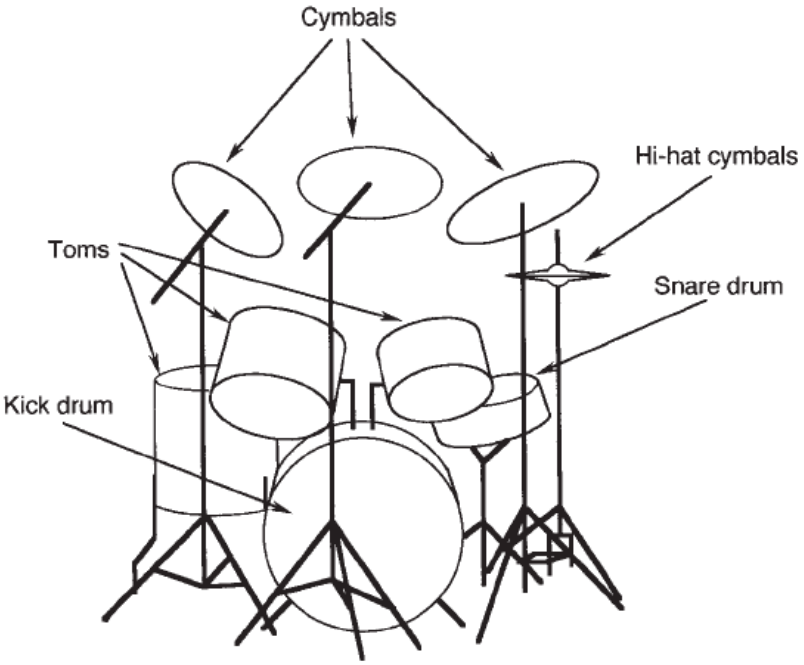


Figura 4.10: Set de percusión típico usado en las tomas de batería

Una vez expuesto el método seguido en las tomas de la batería, podemos pasar a mostrar los resultados obtenidos.

En el caso de las pistas de batería correspondientes al grupo *repium* se muestran a continuación tres casos, en los los que los dos primeros son bucles o *loops* creados a partir de fragmentos de una pista y el último es una pista completa de batería. Los loops se hacen para hacer más significativas las diferencias entre la suma alineada y la no alineada, y así hacerlo más apreciable.

Loop de caja y hit-hat

Aquí se ha extraido un golpe de caja sobre dos golpes de hit-hat cerrado y se ha repetido durante 15 segundos creando el *loop*. Una vez hecho, se han introducido en el software desarrollado las tomas del micrófono OHR y de caja, para analizar el desfase, dando como resultado el mostrado en la figura 4.11. Además de saber el retraso en muestras

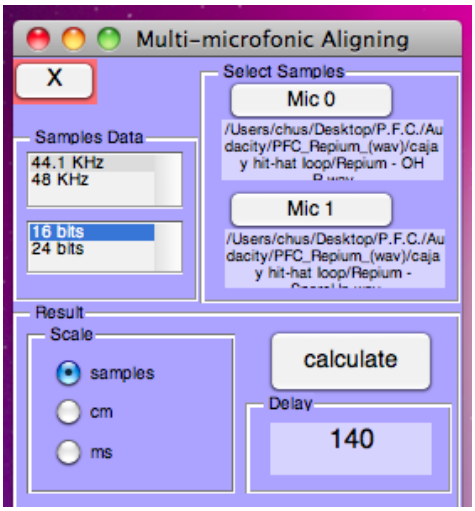


Figura 4.11: Retraso en muestras entre OHR y el micrófono de caja

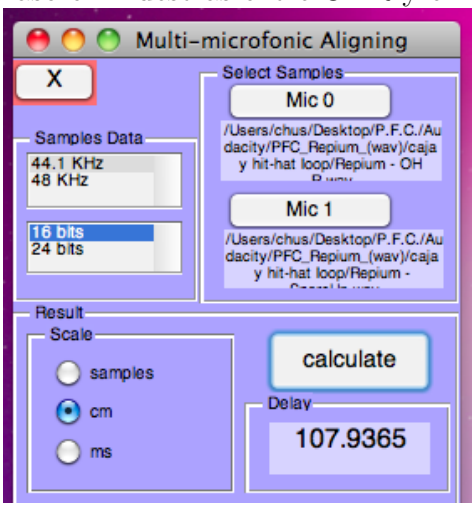


Figura 4.12: Retraso en centímetros entre OHR y el micrófono de caja

también podemos saber la distancia a la que se encuentran ambos micrófonos, cam-

biando en nuestro software la opción de escala de muestras a centímetros, obteniendo una distancia de 1 metro y 7 centímetros, tal y como indica la imagen 4.12.

Hallado el retraso entre ambas señales y corregido éste, hallemos los espectrogramas de las dos señales resultantes, la suma normal y la suma alineada de las dos señales OHR y de caja. Como muestra la figura 4.13, en el espectrograma de abajo, correspondiente

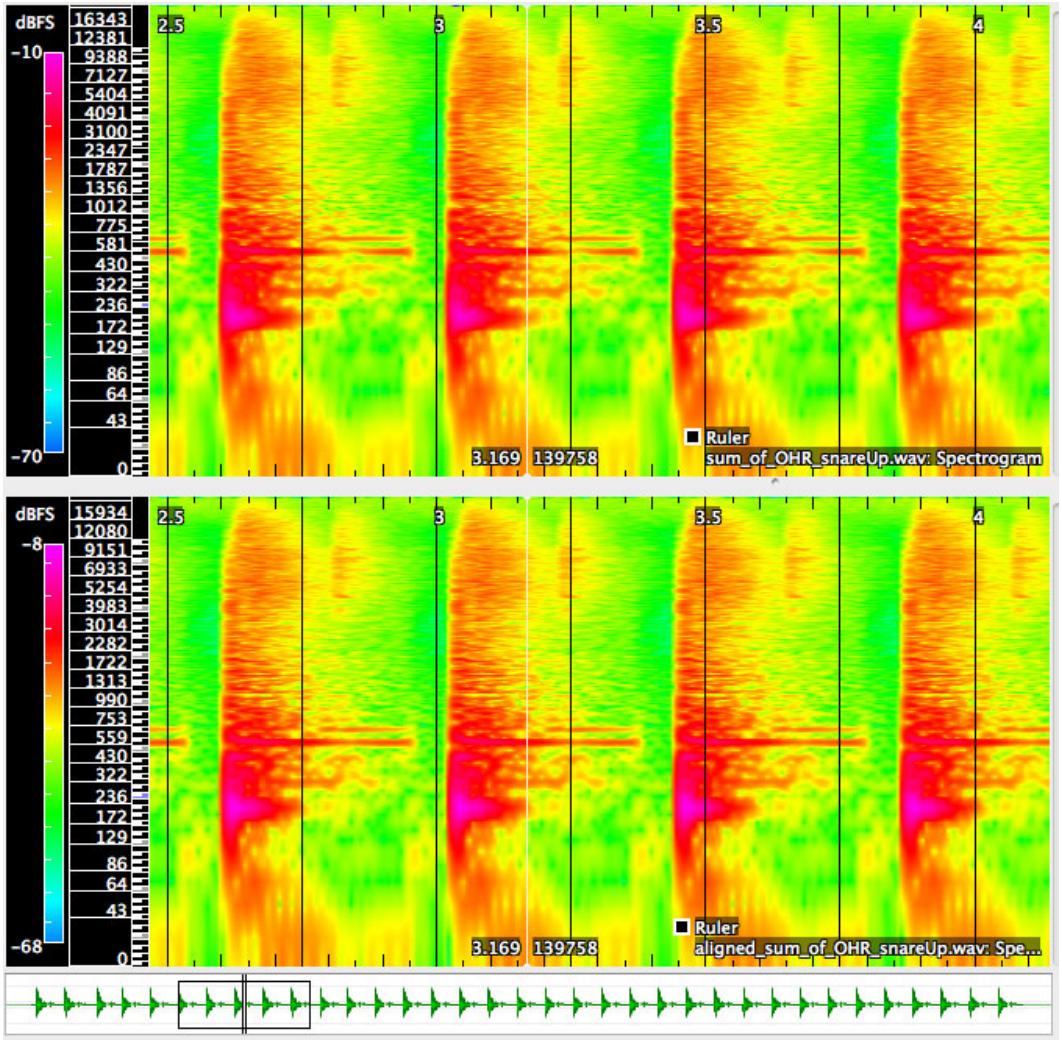


Figura 4.13: Espectrogramas correspondientes a las señales de OHR y caja del loop de caja y hit-hat.

a la suma alineada de las señales de OHR y caja, hay un aumento significativo del nivel

en las frecuencias próximas a los 300-500 Hz.

Loop redoble de caja

Aquí se ha extraído un redoble de caja y se ha formado un fragmento de 15 segundos. Al introducirlo en nuestro software nos da como resultado un retardo de 149 muestras como muestra la imagen 4.14. En el espectro de abajo de la figura 4.16 se puede observar

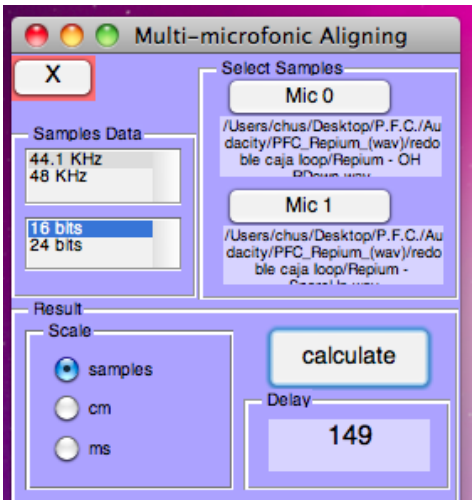


Figura 4.14: Retraso en muestras entre OHR y el micrófono de caja

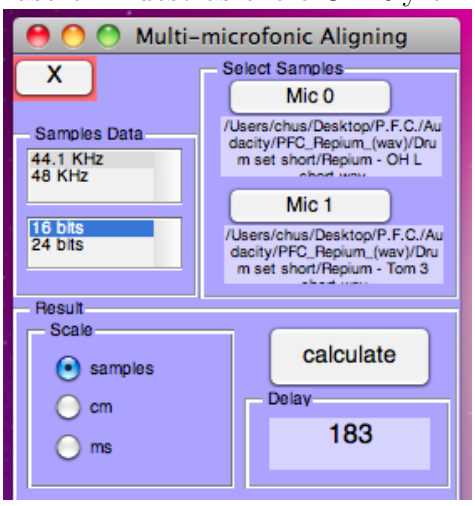


Figura 4.15: Retraso en muestras entre OHR y el micrófono del timbal base o goliat.

como hay un aumento más acentuado incluso, que el apreciado en el caso anterior. Este

se concentra más en las frecuencias colindantes a 200 Hz, zona marcada en el espectro por un intenso rosa.

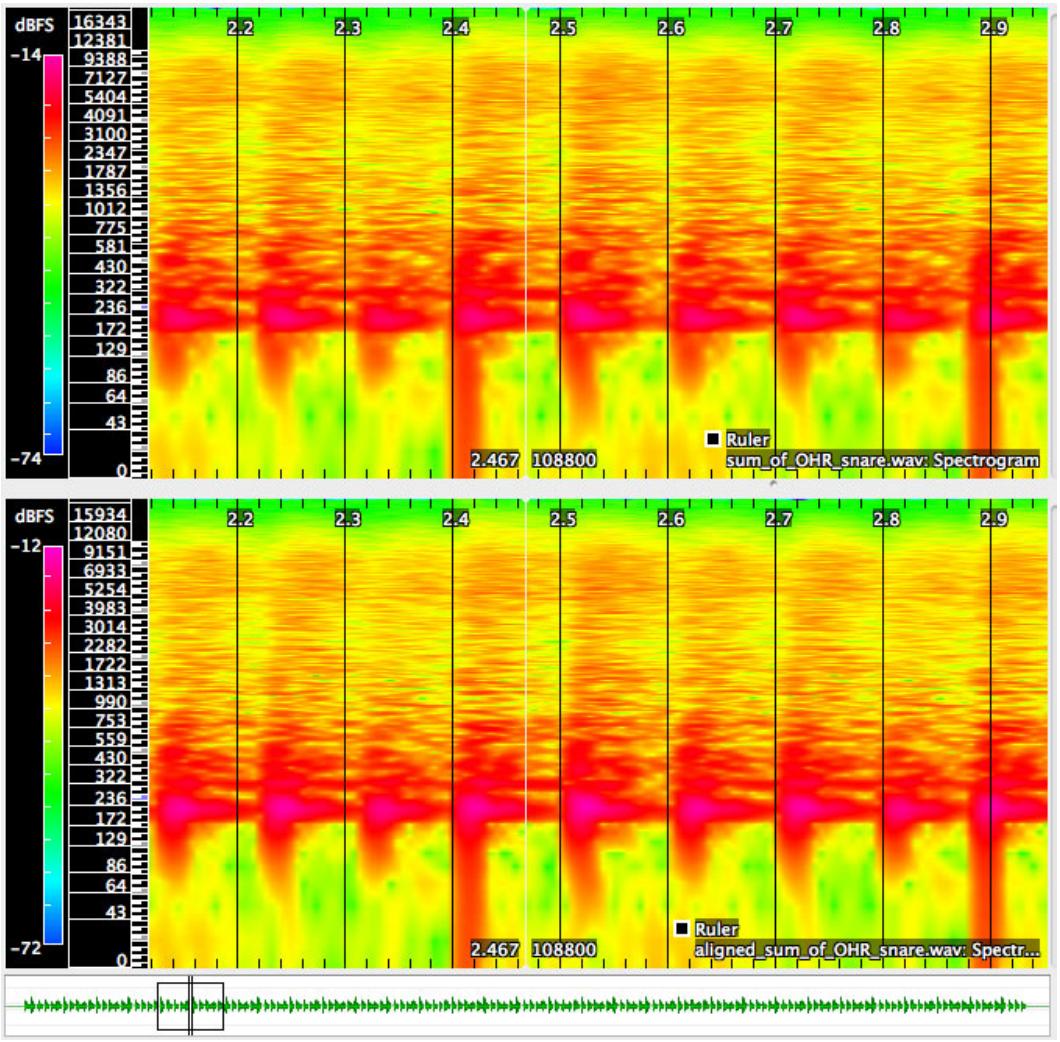


Figura 4.16: Espectrogramas correspondientes a las señales de OHR y caja del loop de redoble de caja.

Pista completa de batería

Aquí se ha dejado la pista de batería tal y como la tendría el técnico del estudio en su ordenador, y se han analizado, con el software desarrollado, los micrófonos de OHR

con el timbal base o goliat, dando un retraso de 183 muestras (imagen 4.15).

En el espectrograma de la suma alineada, mostrado en la parte inferior de la imagen 4.17 se observa como, del mismo modo que el caso anterior, hay un aumento del nivel entorno a 200 Hz, pero en este caso llega casi hasta los 80 Hz, debido también a que el instrumento en cuestión es el más grave de los tres timbales.

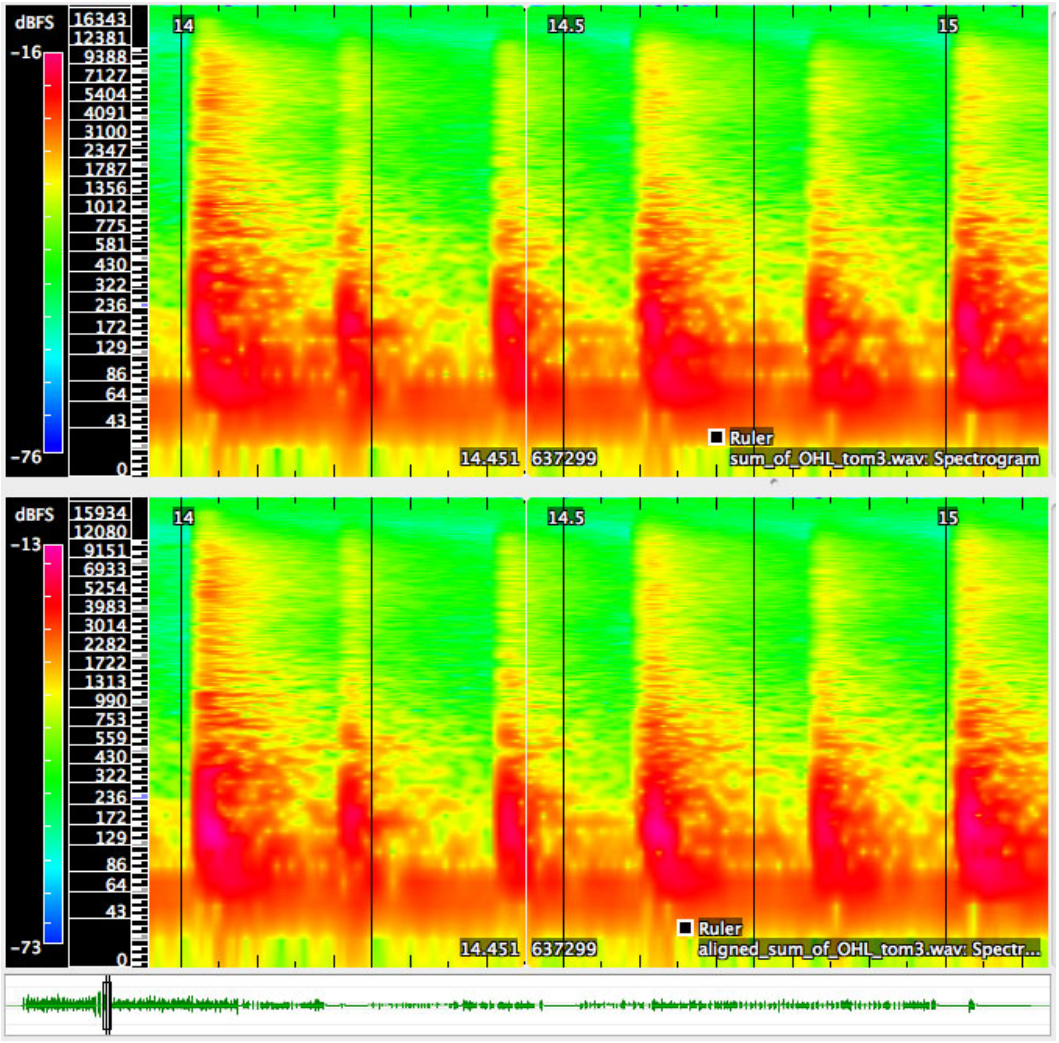


Figura 4.17: Espectrogramas correspondientes a las señales de OHR y timbal base o goliat.

4.2.2. Alineamiento en las tomas de Bajo

En cuanto a las tomas de bajo, se ha utilizado un bajo eléctrico con su amplificador, al cual se le pasa la señal del bajo, pero ésta pasa antes por una *DI box* o caja de inyección, como muestra la imagen 4.18, donde la señal de bajo sale hacia el amplificador y hacia la mesa de grabación, teniendo así dos tomas resultantes, la del propio bajo directamente, y la del bajo pasando por el amplificador.

El hecho de tener una señal eléctrica (la señal directa del bajo) y otra señal acústica

Figura 4.18: Técnica seguida para la grabación de las pistas de bajo.

recogida por un micrófono (la señal del amplificador), nos puede inducir serias dudas sobre el alineamiento de éstas, cosa que comprobaremos analizando ambas señales obtenidas de la grabación del bajo con el software desarrollado.

En primer lugar se muestra el análisis de las señales de bajo obtenidas de una grabación

de un tema de Queen, en las que comprobamos si realmente existe ese desfase que imaginábamos entre las señales de la DI y la señal del amplificador.

Como se observa en la imagen 4.19 realmente existe ese retardo, siendo en este caso de 87 muestras.

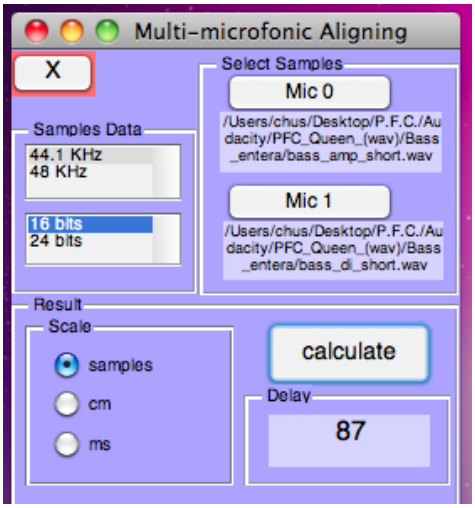


Figura 4.19: Retraso en muestras entre la señal directa del bajo y la señal del amplificador de la toma de *Queen*.

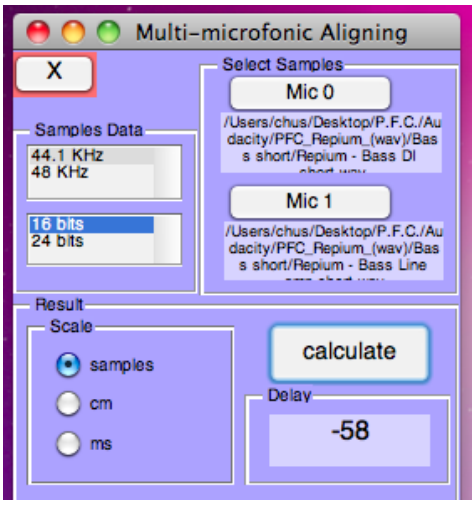


Figura 4.20: Retraso en muestras entre la señal directa del bajo y la señal del amplificador de la toma de *Repium*.

Esto tiene efectos audibles a la hora de sumar ambas señales, cosa que muchos de los técnicos pasan por alto cuando hacen su mezcla. Estos efectos pueden observarse en los espectrogramas de la imgen 4.21. Como se puede apreciar, las frecuencias contiguas a 100 Hz (llegando hasta los 60 Hz por debajo y hasta los 200 Hz por arriba) se ven acentuadas gracias a la alineación de las señales.

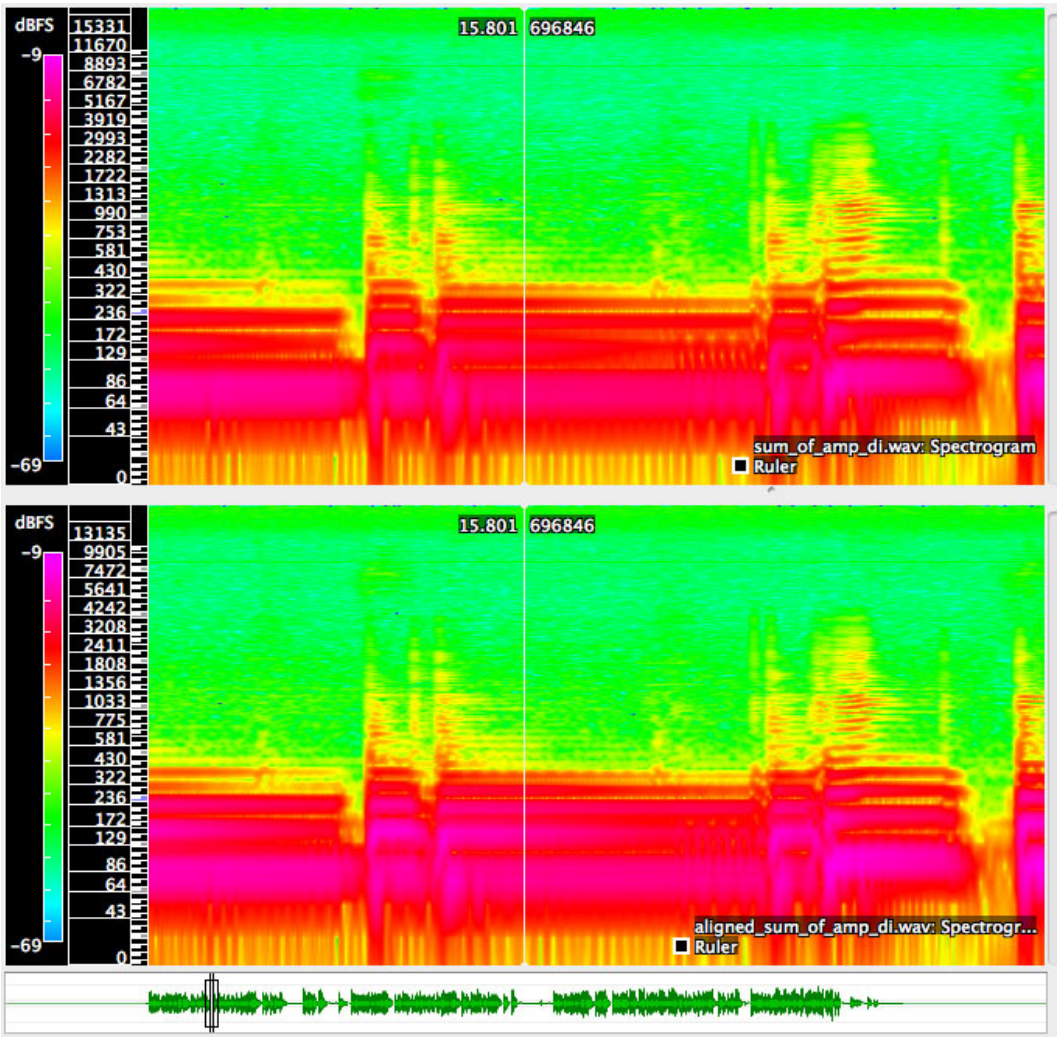


Figura 4.21: Espectrogramas correspondientes a la señal de bajo directa y la señal del amplificador de la toma de *Queen*.

A continuación analizamos las señales de bajo obtenidas de la grabación en el estudio del grupo *Repium*. El software diseñado nos muestra un retardo de 58 muestras (imagen 4.20) al analizar ambas señales. Una vez alineadas, obtenemos los espectrogramas de la suma normal y la suma alineada de las dos señales.

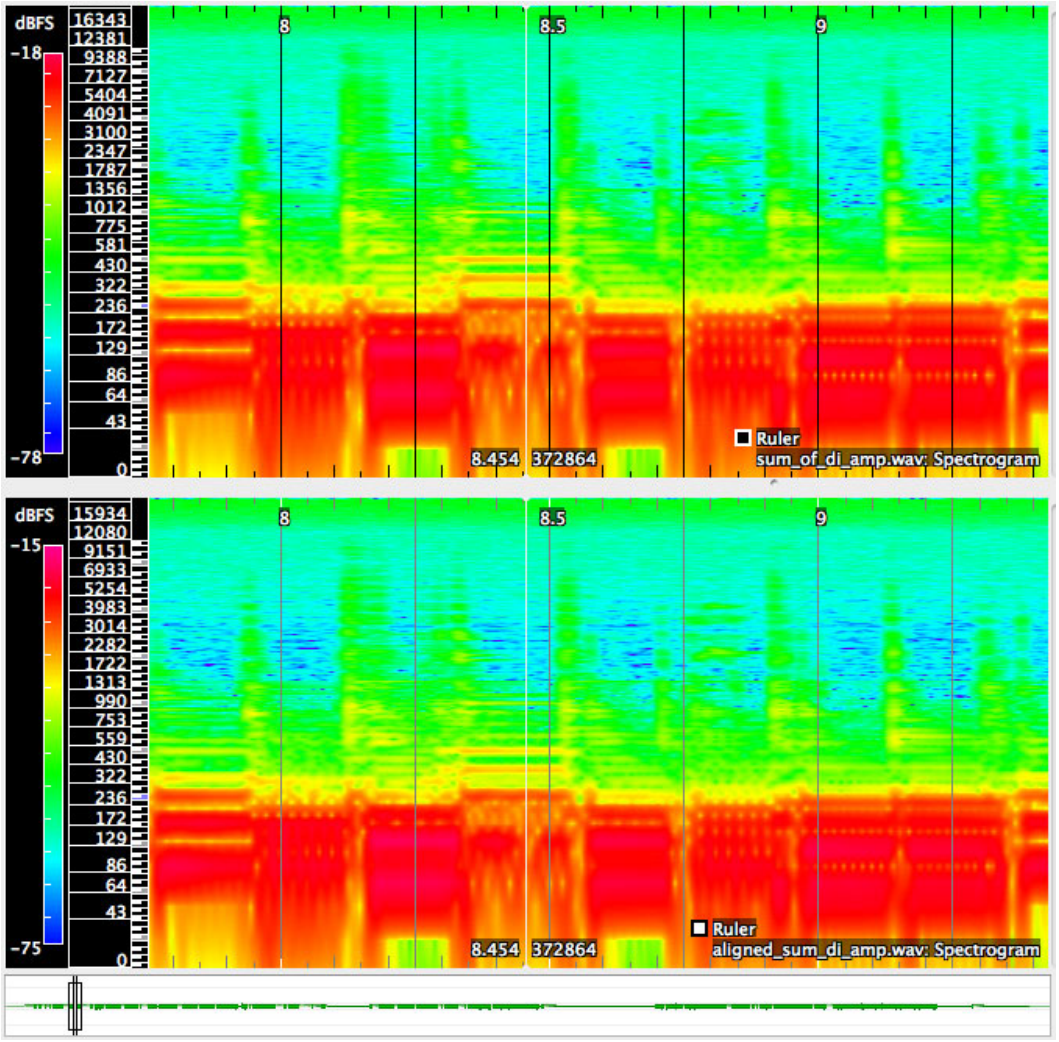


Figura 4.22: Espectrogramas correspondientes a la señal de bajo directa y la señal del amplificador de la toma de *Repium*.

Como se puede apreciar en la imagen 4.22, gracias a la alineación las frecuencias entorno a 80 Hz (llegando a 172 Hz por arriba y casi a 40Hz) se acentúan pero también

observamos como las frecuencias altas quedan mas limpias, es decir, cuando suena la nota la energía se concentra más alrededor de la frecuencia correspondiente a la nota, y no se dispersa por el resto del espectro de frecuencias.

4.2.3. Alineamiento en las tomas de guitarra

Respecto a las tomas de guitarra, se ha utilizado una técnica muy común a la hora de grabar guitarras con su amplificador. esto se hace debido a que el sonido de cada músico suele ir ligado al de su amplificador, es como una extensión más de su guitarra, amplificador y guitarra sacan el sonido característico del propio músico.

Esta técnica consiste en grabar el sonido que sale del amplificador mediante dos

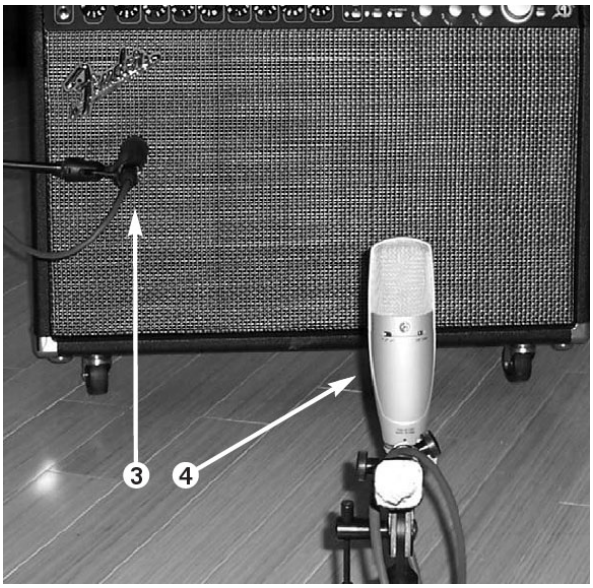


Figura 4.23: Técnica seguida para la grabación de las pistas de guitarra.

micrófonos, normalmente un micrófono dinámico pegado al cono, perpendicular a éste y separado del eje del mismo (número 3 en la imagen 4.23), y un micrófono de condensador y diafragma ancho un poco separado del cono apuntando justo al centro de éste (número 4 en la figura 4.23).

En el caso de nuestra grabación se han utilizado como micrófono dinámico un Shure 57 y como micrófono de condensador un AKG 414.

En el siguiente ejemplo se muestra la guitarra rítmica del grupo *Repium*. Al analizar las muestras de ambos micrófonos con el software diseñado observamos como esta se-

paración (imagen 4.24) induce un retraso en muestras (imagen 4.25) que supondrá un efecto audible en la suma no alineada de ambas señales por parte del técnico a la hora de hacer la mezcla.

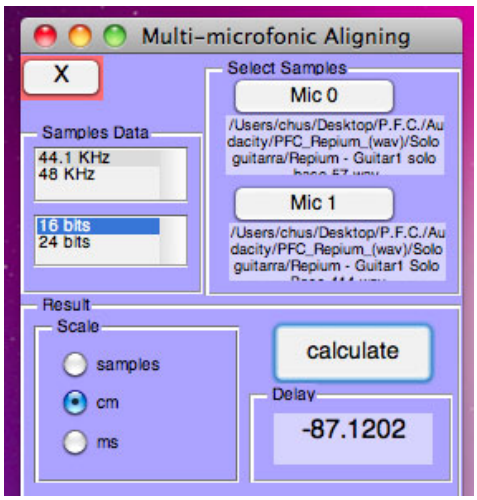


Figura 4.24: Separación en centímetros entre el micrófono AKG 414 y el Shure 57 de la toma de guitarra rítmica de *Repium*.

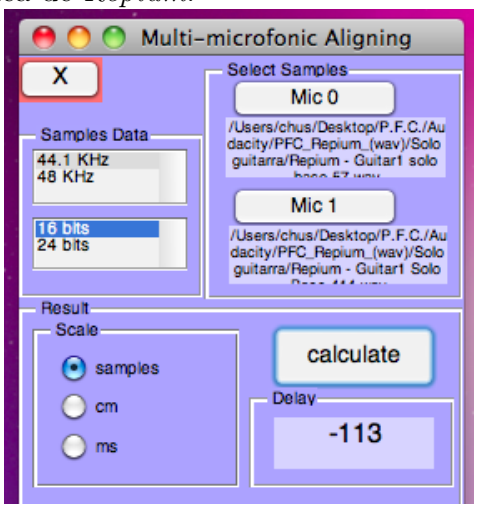


Figura 4.25: Retraso en muestras entre la señal del micrófono AKG 414 y del Shure 57 de la toma de guitarra rítmica de *Repium*.

Este efecto se ve patente en el espectrograma de la imagen 4.26, donde la energía se

concentra entre 100 y 500 Hz quedando el resto del espectro un poco más limpio.

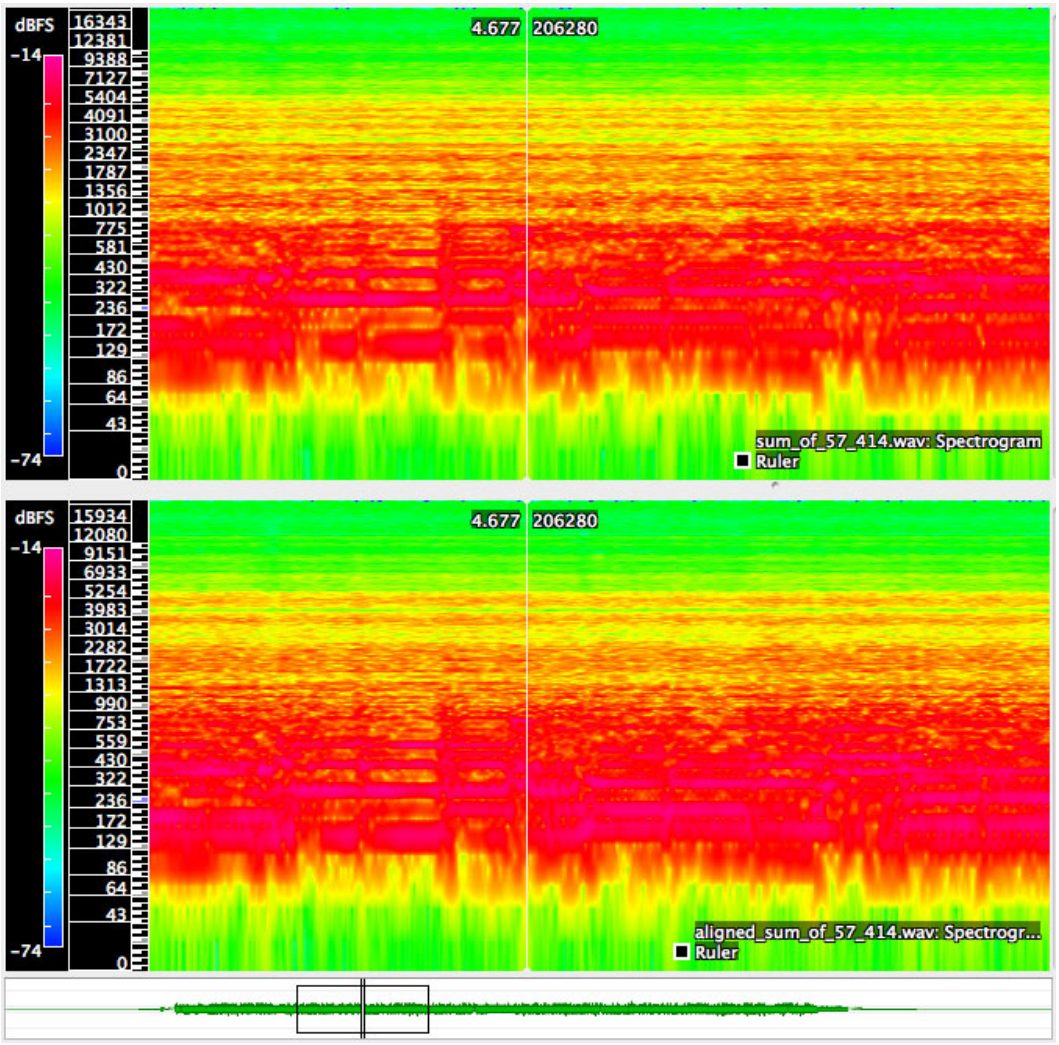


Figura 4.26: Espectrogramas correspondientes a la señal de guitarra rítmica de *Repium*.

En el caso de la guitarra solista también se ha grabado siguiendo esta técnica y como se puede apreciar el la imagen 4.27 también existe un retraso entre las señales de los micrófonos. Si observamos el espectrograma de la suma de las señales de los dos micrófonos (imagen 4.28) podemos observar que, al igual que el caso de la guitarra

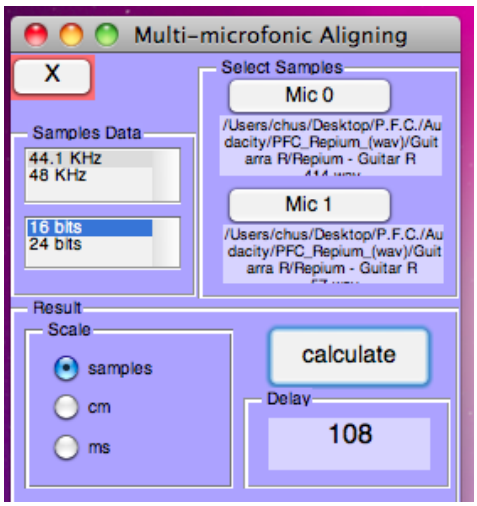


Figura 4.27: Retraso en muestras entre la señal del micrófono AKG 414 y del Shure 57 de la toma de guitarra solista de *Repium*.

rítmica, se acentúan las frecuencias comprendidas entre 100 y 500 Hz al concentrarse más la energía en ese margen de frecuencias.

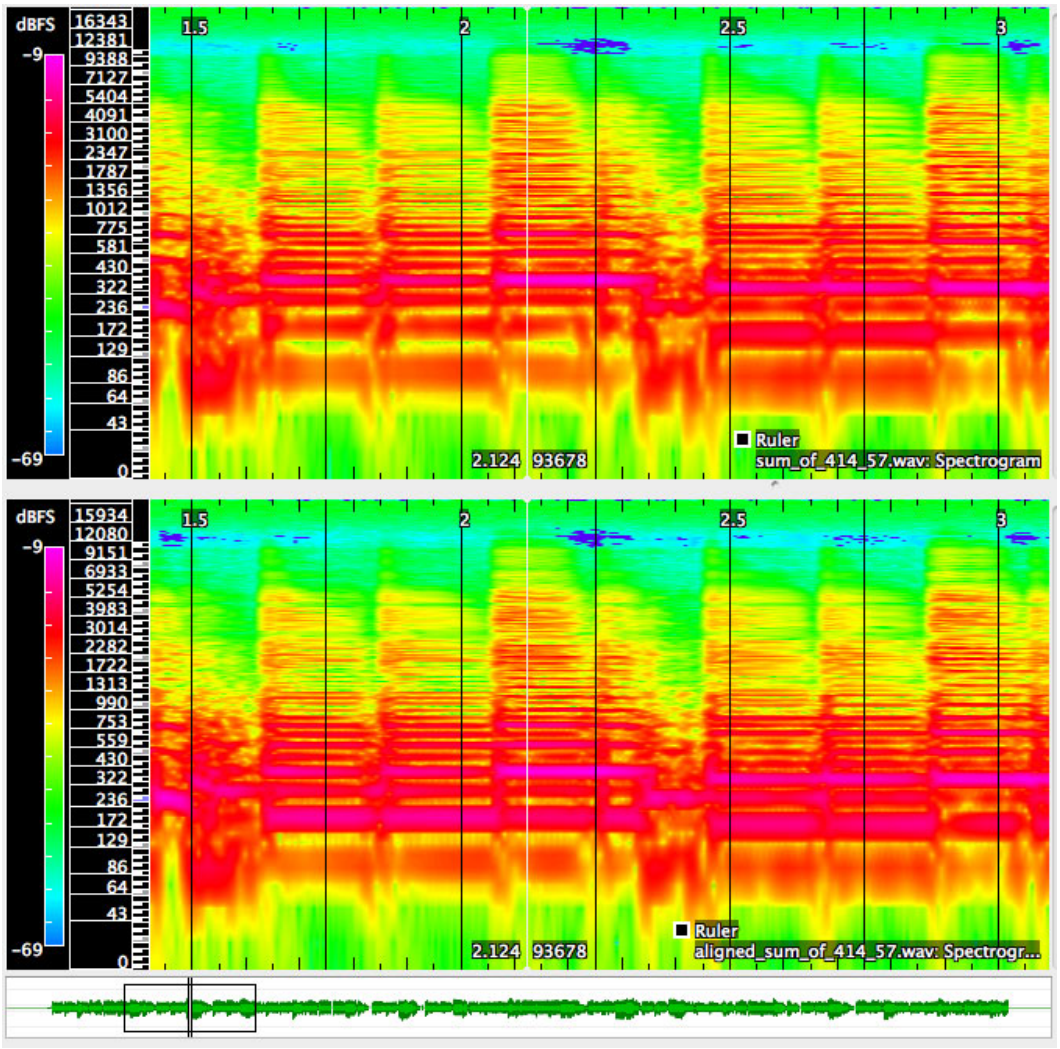


Figura 4.28: Espectrogramas correspondientes a la señal de guitarra solista de *Repium*.

4.3. Grabaciones fuera del estudio

Para esta última sección, en primer lugar se presentan los resultados obtenidos de la grabación de una orquesta de música clásica de 50 componentes. Ante la imposibilidad de grabar a una orquesta sinfónica completa, me vi en la necesidad de recurrir a una orquesta de jóvenes talentos, que a pesar de ser estudiantes de últimos cursos de conservatorio, se comportaron como auténticos músicos profesionales durante la sesión de grabación.
En segundo lugar, se muestran los resultados obtenidos de la grabación de un conjunto coral en el interior de una iglesia románica, formado por cuatro voces (bajo, tenor, alto y soprano).

4.3.1. Alineamiento en la grabación de una orquesta

La orquesta se llama *Joven Orquesta Sinfónica Autónoma de Cantabria* y la grabación tuvo lugar en un aula, donde habitualmente ensaya una coral. Las condiciones no eran las idóneas para una grabación de estas características, pero se aprecia el fenómeno de estudio de este proyecto.
La técnica microfónica escogida fue la colocación de un par micrófonico estéreo XY (Rode NT-5) sobre el director, al que a partir de ahora denominaremos par principal. Para la captación de la sección de vientos, se utilizaron dos micrófonos (Audio-Technica ES935 ML6) bastante directivos, para captar solo la zona donde se coloquen.En este caso se pusieron dos, para captar toda la zona de viento madera y viento metal, aproximadamente en el mismo eje que el par principal (L y R). La disposición de los músicos en el aula, así como su director y la colocación de los micrófonos se pueden ver en la imagen 4.29 y 4.30. Según esta disposición microfónica se grabó la sesión de ensayo de la orquesta. A continuación se muestran dos fragmentos de dos temas interpretados.
El primero de ellos pertenece a la *Fantasía Romeo y Julieta* de P.I. Tchaikovsky, más

Figura 4.29: Distribución de los micrófonos para la grabación de la orquesta.

Figura 4.30: Distribución de los micrófonos para la grabación de la orquesta

concretamente al movimiento *allegro*. De las cuatro pistas disponibles se han elegido para analizar con el software diseñado la toma L del par principal y el micrófono puntual de la Izquierda (para mantener el eje en ambos micrófonos). El resultado obtenido tras analizar ambas muestras es un desfase de 1829 muestras, tal y como indica la imagen 4.31. Tras conocer el desfase pasamos a visualizar las señales sumadas y guar-

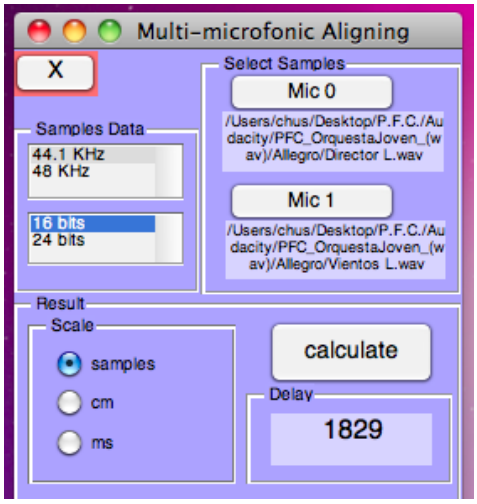


Figura 4.31: Resultado del análisis de las señales de los micrófonos de la izquierda.

dadas mediante el propio software, de modo que mediante los espectrogramas podamos apreciar las diferencias entre la suma normal y la suma alineada de las señales de los micófonos. Esto se comprueba en la imagen 4.32, donde se observa como se han reforzado las frecuencias medias bajas, concentrándose ahí la energía dispersa por las frecuencias altas.

Este segundo fragmento pertenece al *Danzón Cubano n2* de Arturo Márquez. Aquí también se han cogido los micrófonos de la izquierda, tanto el par principal como el puntual, por lo que pasaremos a mostrar directamente el espectrograma, pues permanece la misma distribución microfónica. En dicho espectrograma (imagen 4.33), se observa como

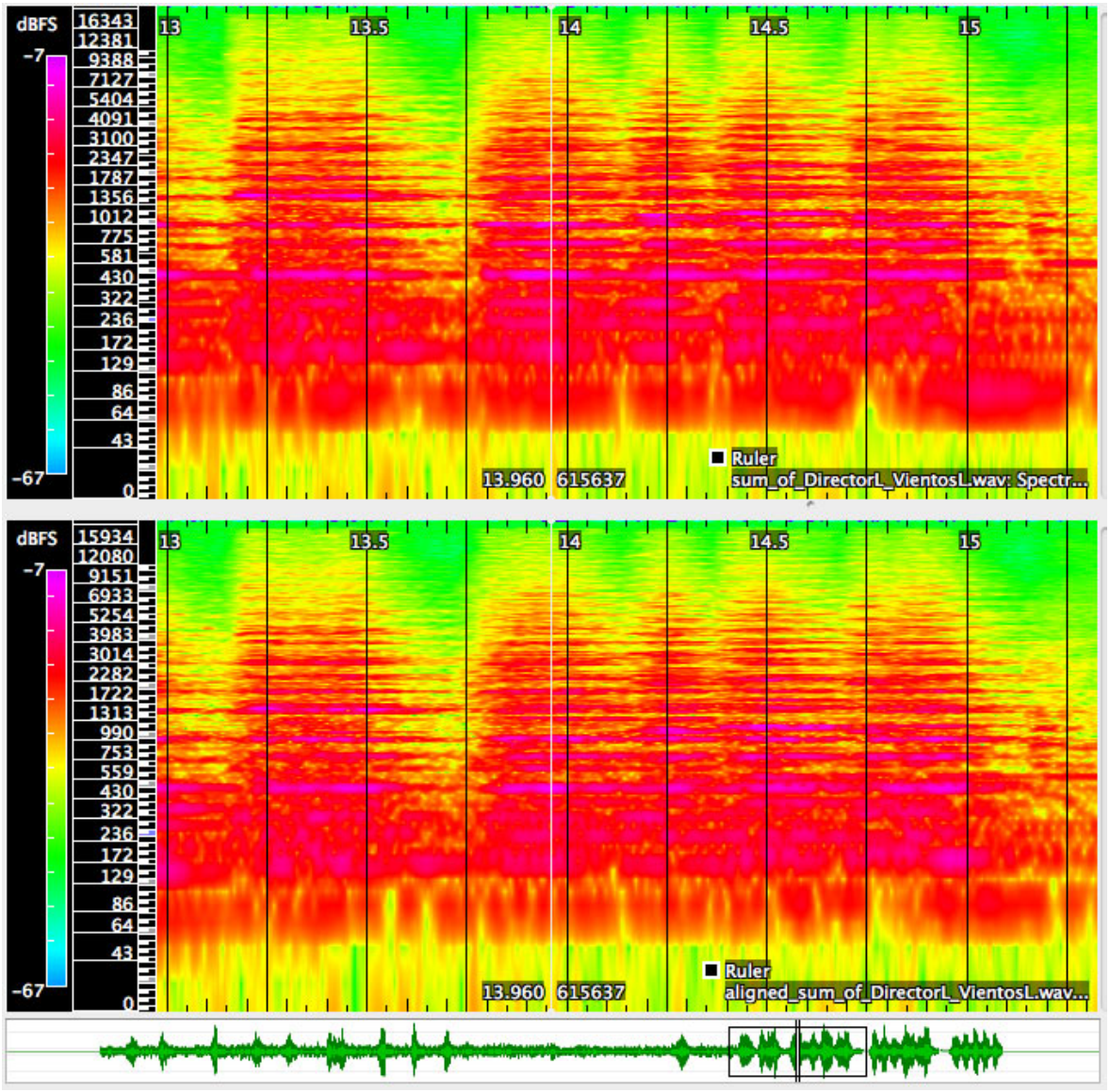


Figura 4.32: Espectrogramas correspondientes a las señales del par principal izquierdo y el puntual izquierdo del *allegro*.

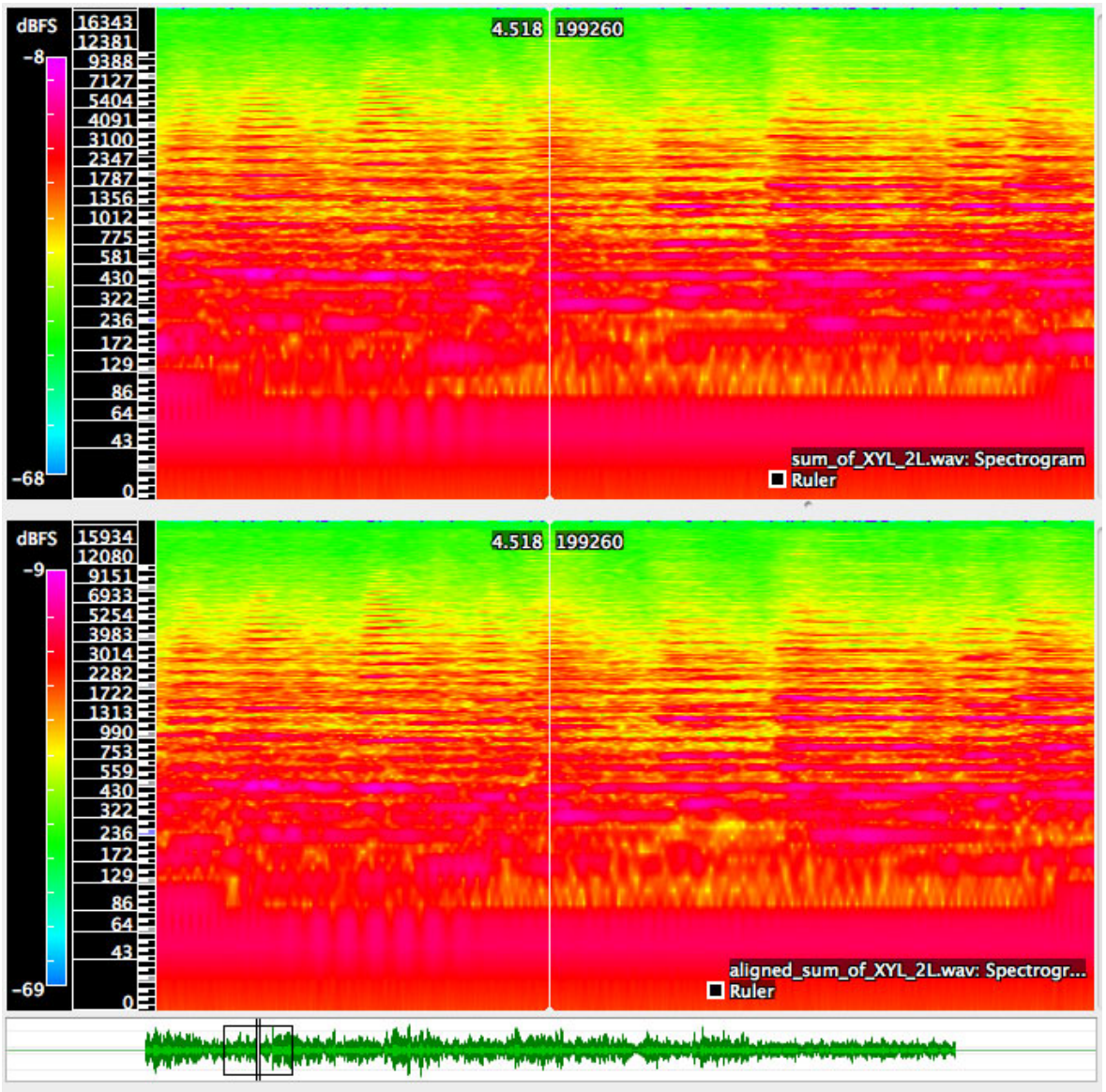


Figura 4.33: Espectrogramas correspondientes a las señales del par principal izquierdo y el puntual izquierdo del *Danzón Cubano*.

4.3.2. Alineamiento en la grabación de una agrupación coral

La agrupación coral grabada se llama *Commentos Vocis* y está compuesta de cuatro cuerdas: Héctor Guerrero (bajo y director), Helia Martínez (alto), Miguel Bernal (tenor), Celia Alcedo (soprano), además hay un actor que interpreta unos poemas y del que también se presentará una grabación.

Antes de nada me gustaría agradecer la enorme generosidad de Armando Gutiérrez, técnico de sonido de Radio Nacional de España en la delegación de Cantabria, ya que coincidimos trabajando en este evento, por mi parte en la sonorización, y por su parte en la grabación de la agrupación coral para Radio Nacional de España central.

Tras una larga charla explicativa sobre la realización de este proyecto como Proyecto Final de mi Carrera, le comenté si podía incluir un sexto micrófono ademas de los cuatro destinados a las voces y el quinto para el actor. Accedió gustosamente, y coloqué este sexto micrófono en la parte final de la iglesia, como se puede apreciar en la imagen 4.34, justo delante del altillo del coro parroquial.

De esta forma el micrófono captaría la reverberación natural de la sala (en este caso una Iglesia románica) y mezclando esta señal con la de los otros cuatro micrófonos tendríamos una muy buena mezcla sin necesidad de añadir efectos a las tomas de las cuatro voces.

La disposición en la escena es simple, tal y como se muestra en la imagen 4.35. Se sitúan los cuatro componentes sobre el escenario formando una media luna delante del altar de la iglesia, delante de cada micrófono. Al fondo, si encuentra el sexto micrófono.

La sensación de profundidad que se aprecia en la imagen 4.36 queda de esta forma muy patente en las grabaciones realizadas.

Figura 4.34: Disposición del sexto micrófono en la parte posterior de la iglesia.

Figura 4.35: Distribución de los micrófonos sobre el escenario para la grabación de la agrupación coral.

Figura 4.36: Distribución de los micrófonos en la Iglesia para la grabación de la agrupación coral.

Empezamos mostrando un fragmento de los poemas recitados por el actor. Como se aprecia en la imagen 4.36 el concierto en un principio iba a sonorizarse mediante dos cajas autoamplificadas de 400 Watios cada una, pero tras la prueba de los músicos, éstos decidieron relegar la sonorización única y exclusivamente para las partes recitadas por el actor, con lo cual se situó un micrófono dinámico (Shure 58) muy cerca del actor dedicado a la sonorización y un micrófono de condensador dedicado a la grabación separado unos centímetros, como podemos apreciar en la imagen 4.37.

Ésto influye en la medición del retardo, ya que los altavoces lanzan el sonido con

Figura 4.37: Localización del micrófono empleado para la sonorización del actor.

mucha más potencia hacia el fondo de la Iglesia y el micrófono que recoge la señal que se manda a éstos está mucho más próximo al actor que el micrófono utilizado para la grabación. Por ello el resultado de la medición del retardo entre el micrófono del presentador y el micrófono del fondo de la iglesia (imagen 4.38) es mucho menor que el retardo entre los micrófonos de los vocalistas y el del fondo de la Iglesia. Si vamos al espectrograma obtenido de la sumas alineadas y no alineadas de las señales de estos

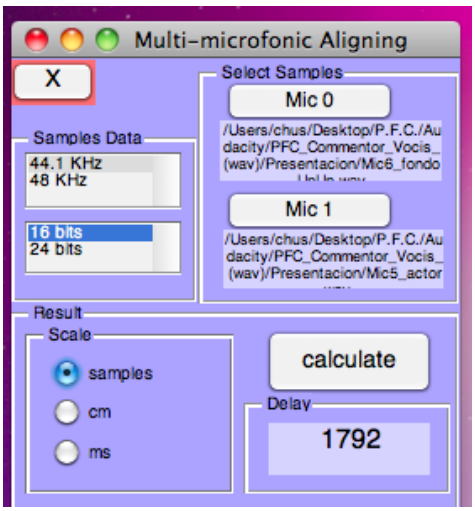


Figura 4.38: Retraso en muestras entre la señal del micrófono del actor y el situado al fondo de la Iglesia.

micrófonos (imagen 4.39), podemos observar como se concentra la energía sobre las bajas y medias frecuencias, liberando en gran medida a las frecuencias superiores a 1 KHz, frecuencias sobre todo compuestas por reflexiones y armónicos.

A continuación se muestran en la imagen 4.3.2 los resultados del análisis, mediante al software diseñado, de cada uno de los micrófonos de los intérpretes con el micrófono del fondo. Se observan pequeñas diferencias debido a la colocación de los micrófonos, ya que no está todos colocados a la misma distancia del sujeto, además los propios sujetos poseen libertad de movimiento, con lo cual no permanecen inmóviles a la misma distancia todo el tiempo. Cabe destacar la diferencia tan abultada del bajo (Imagen 4.40(d)) con el resto de las voces, que es aproximadamente la mitad que el resto. Esto es debido a la presencia de un micrófono dedicado a la sonorización electroacústica del actor, como hemos explicado en este mismo apartado anteriormente. El bajo se encuentra situado muy próximo a este micrófono, lo que hace que se realce y llegue antes al micrófono del fondo.

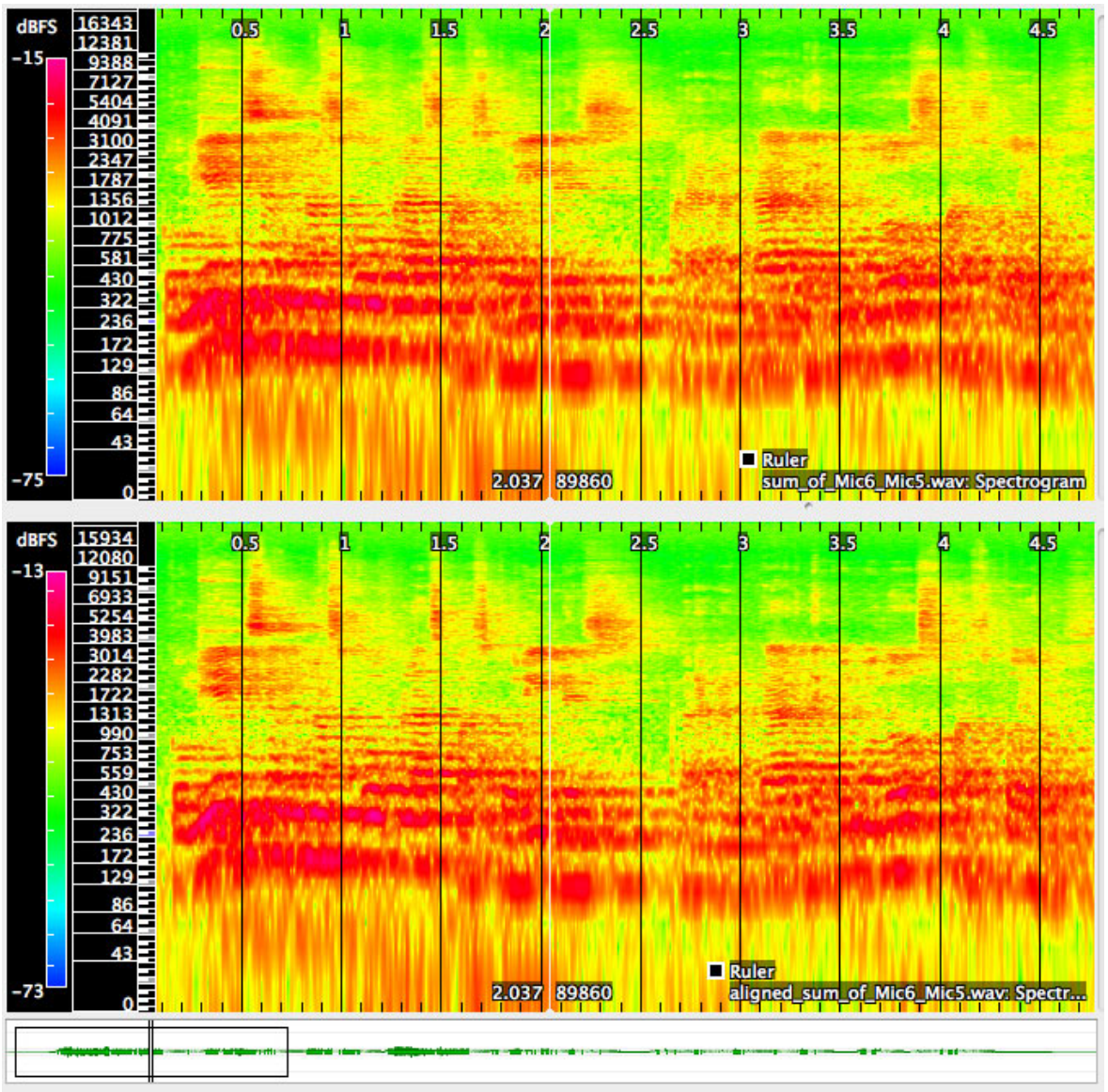
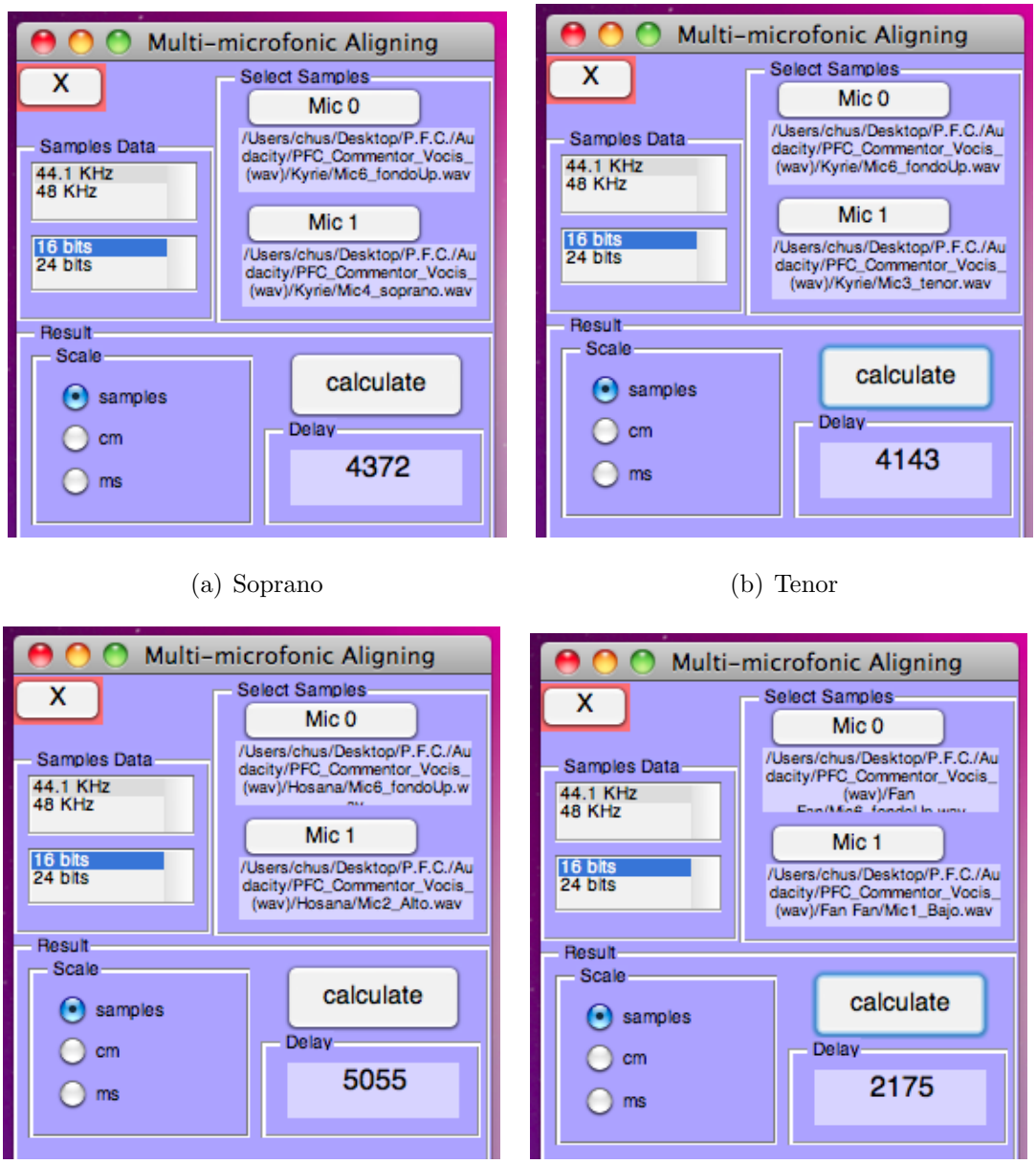


Figura 4.39: Espectrogramas correspondientes a las señales de los micrófonos del actor y del fondo interpretando uno de los poemas



(a) Soprano (b) Tenor

(c) Alto (d) Bajo

Figura 4.40: Retrasos en muestras de las señales de los micrófonos de los intérpretes y del micrófono del fondo de la Iglesia.

Las diferencias entre la suma alineada y la suma no alineada de las señales de los

micrófonos de los músicos con el la señal del micrófono situado al fondo se hacen muy perceptibles auditivamente. Esto se hace visible gracias a los espectrogramas de dichas señales suma. A continuación se muestran los espectrogramas del micrófono de la voz soprano (imagen 4.41), de la voz tenor (imagen 4.42), de la voz alto (imagen 4.43) y por último de la voz bajo (imagen 4.44).

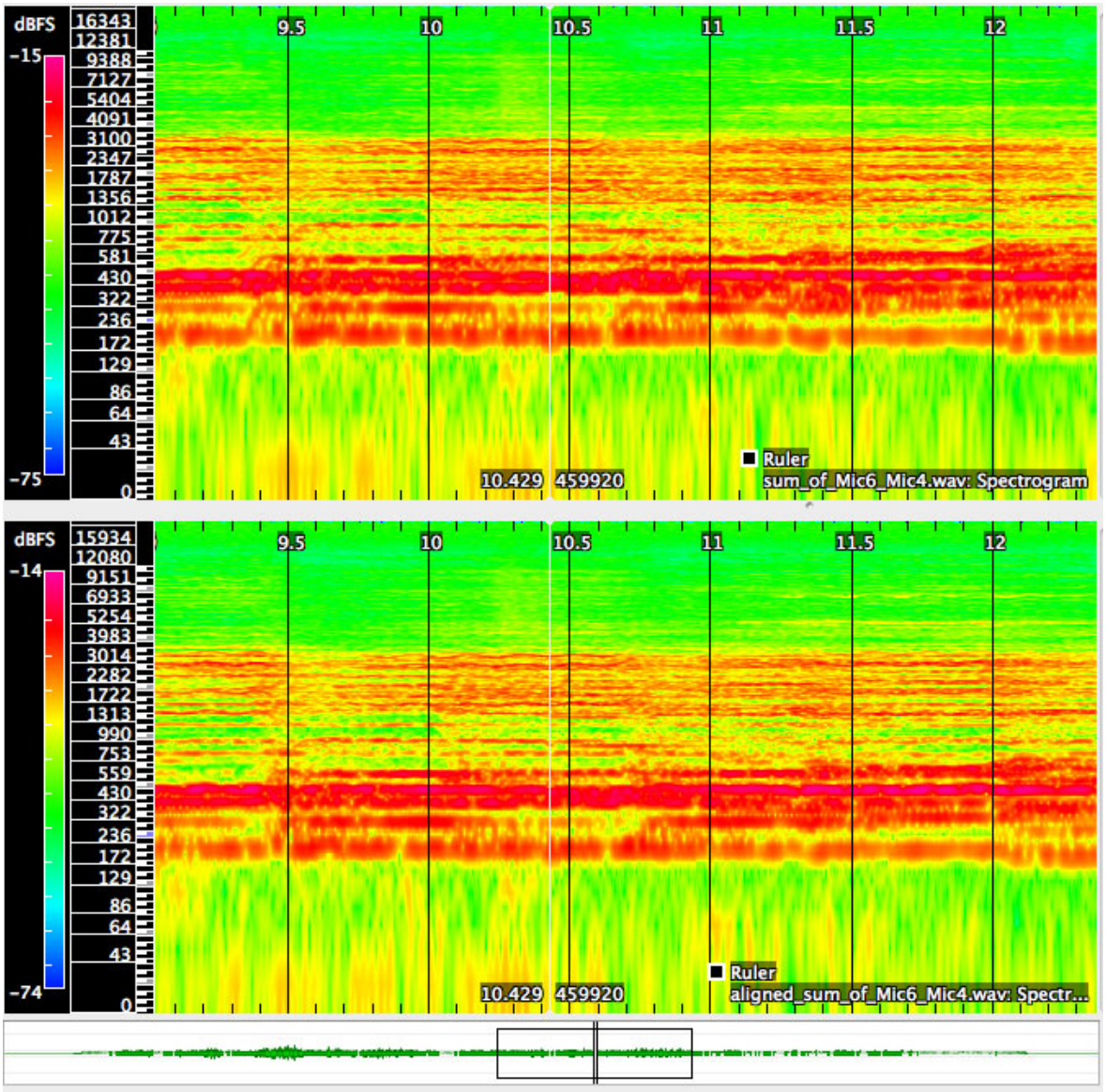


Figura 4.41: Espectrogramas correspondientes a las señales de los micrófonos de la voz soprano y del fondo durante la interpretación del Kyrie

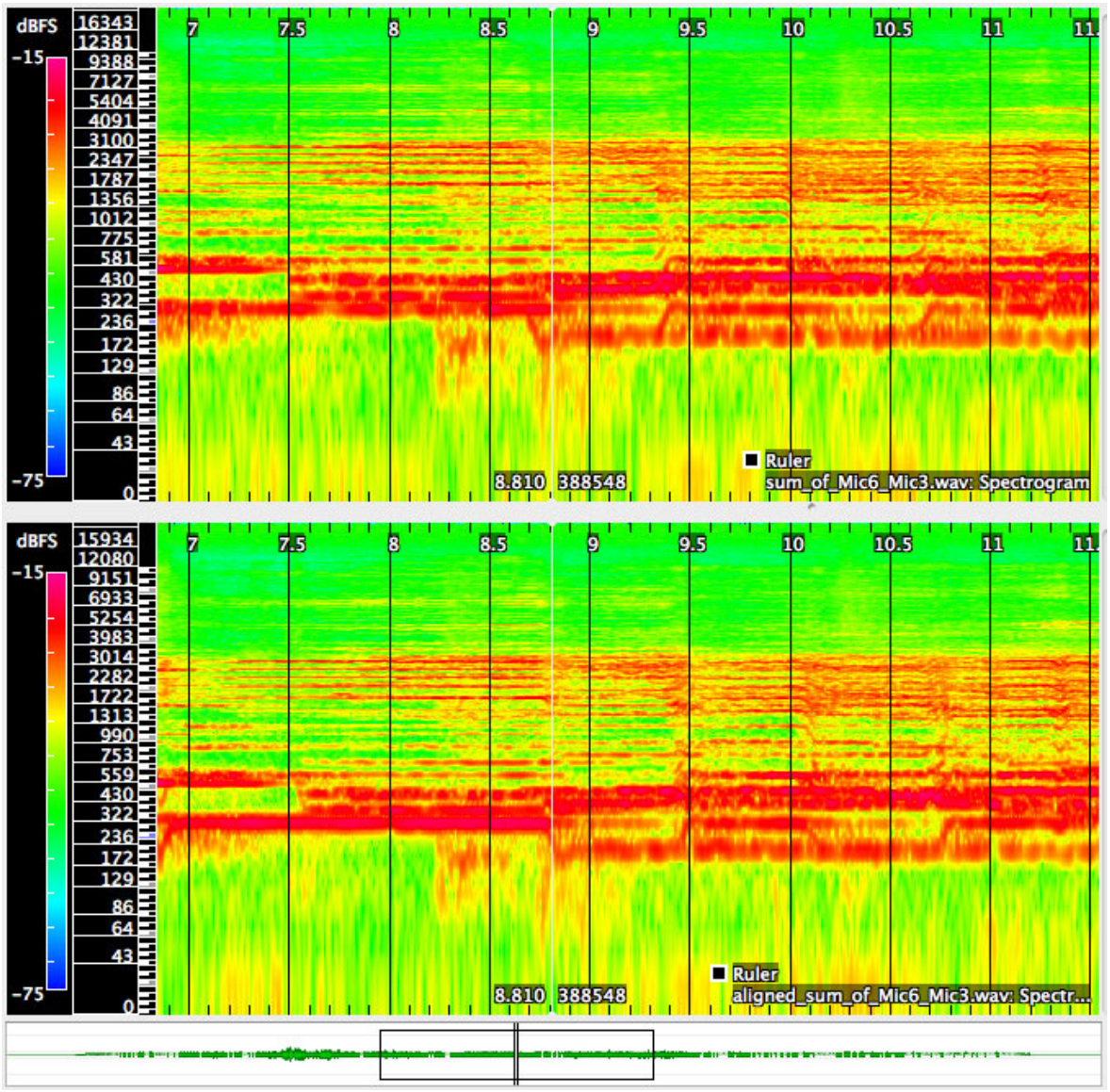


Figura 4.42: Espectrogramas correspondientes a las señales de los micrófonos de la voz tenor y del fondo durante la interpretación del Kyrie

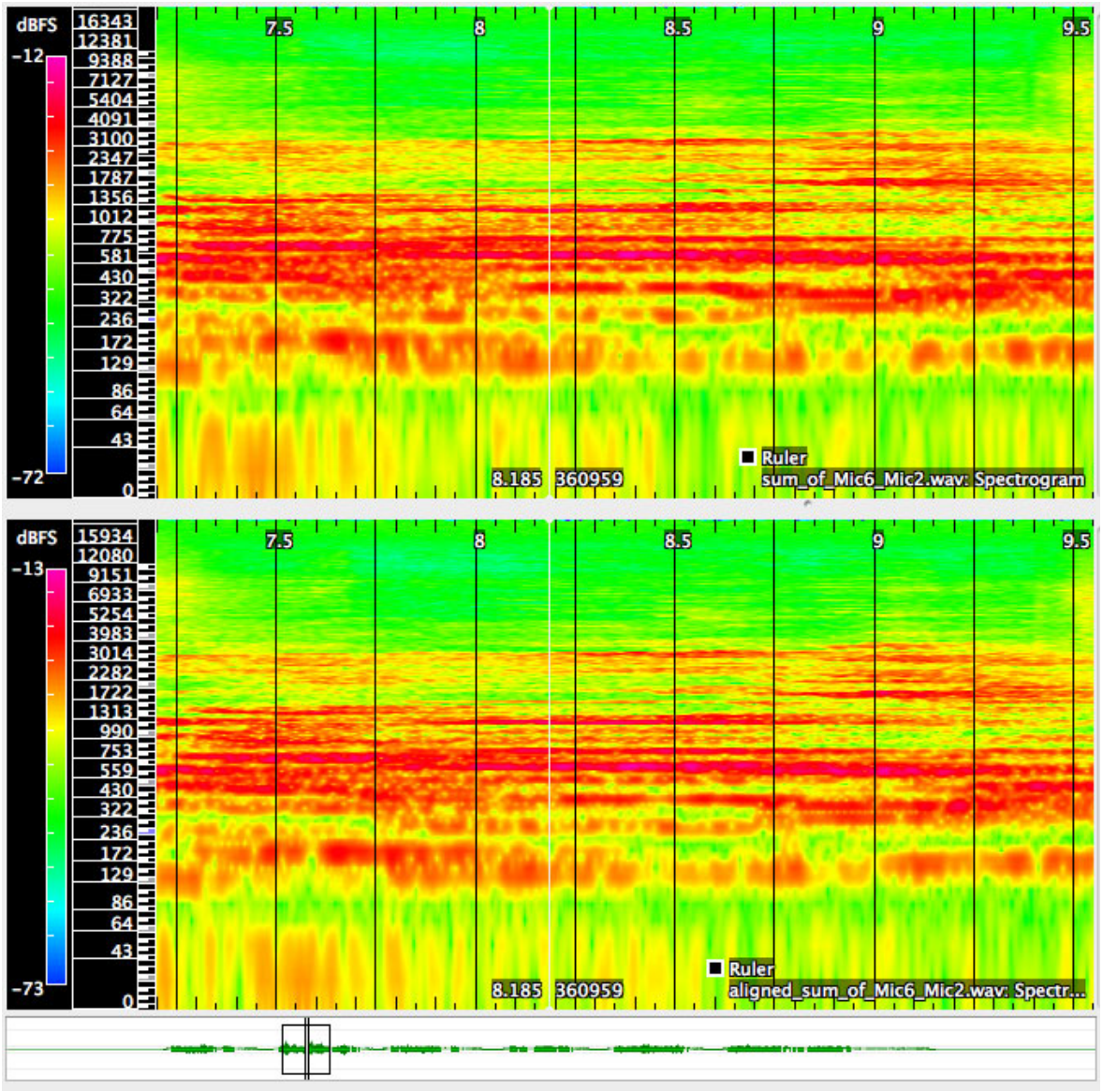


Figura 4.43: Espectrogramas correspondientes a las señales de los micrófonos de la voz alto y del fondo durante la interpretación del Hosana.

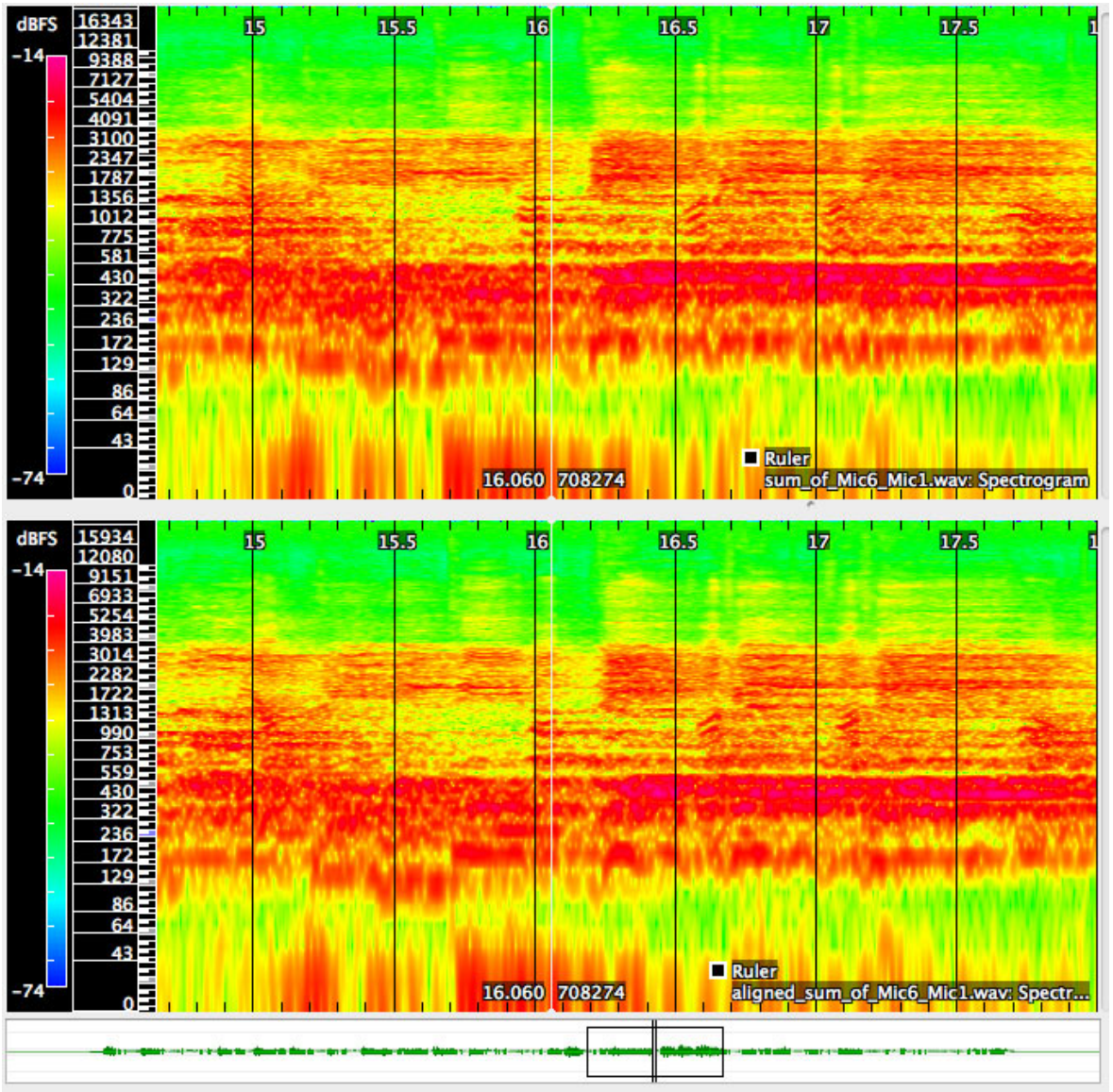


Figura 4.44: Espectrogramas correspondientes a las señales de los micrófonos de la voz bajo y del fondo durante la interpretación del fanfan.

Capítulo 5

Conclusiones.

Grabaciones en estudio

Dentro del estudio podemos resaltar el empleo de este análisis para la sección de percusión. Aquí se observan pros y contras en su utilización.
Por una parte la alineación de los micrófonos empleados para captar el hit-hat, la caja, y los timbales se pueden alinear sin ningún tipo de problema con respecto del par XY situado sobre la cabeza del baterista, situando estos a una distancia aproximadamente igual con respecto de cada micrófono, es decir, estableciendo dos planos imaginarios, uno en el que se encontrasen los primeros micrófonos, y un plano elevado donde se situase el par estéreo.
Por otro lado nos encontramos con el efecto del bombo. Éste al estar situado normalmente su micrófono en el interior y en un plano imaginario inferior al los otros dos planos imaginarios, no se alinea correctamente con el par estéro, ya que los golpes de caja o de timbales (plano imaginario central) llegarán aproximadamente a la vez tanto al micrófono de bombo como al par estéreo, por lo que si establecemos un retraso del micrófono del bombo con respecto del par, se producirá un eco audible de los golpes de caja, cosa que empeoraría nuestra mezcla.

El caso más palpable es el del bajo eléctrico. La alineación de la toma directa con la toma del amplificador debería ser utilizada en todas las grabaciones, ya que siempre existirá un pequeño retraso apreciable auditivamente.

Grabaciones de música clásica

Si las distancias desde el par estéreo principal a las distintas secciones microfónicas de apoyo son más de 4 metros, es una buena idea considerar un retraso de tiempo de

los micrófonos de apoyo. La alineación temporal correcta de los micrófonos de apoyo preservará el timbre de los instrumentos musicales sin colorear con el filtro de peine debido a las diferencias de fase entre los micrófonos de apoyo y el par estéreo principal. Además, una correcta alineación de tiempo será fiel a los sonidos reflejados, lo que dará un registro de información importante sobre el ambiente, es decir, la profundidad, la anchura y la reverberación originada.

La alineación de tiempo depende en gran medida de la habitación o sala de conciertos en los que se lleva a cabo la grabación. Si cada micrófono se retrasa teniendo en cuenta simplemente la distancia y la velocidad del sonido, habrá múltiples problemas de fase si los músicos se mueven durante la reproducción.
Para superar los problemas de fase, y al mismo tiempo conservar el timbre del instrumento individual, debe analizarse las señales de cada micrófono de las secciones de apoyo con respecto del par principal y con esto poder determinar e introducir el retraso correcto. Por supuesto en este retraso se tendría en cuenta las reflexiones en el punto del micrófono de apoyo, lo que nos daría la percepción de la profundidad, tan representativa de la música clásica.

Lineas futuras de trabajo

El futuro de este tipo de análisis de señales recaería en el diseño de *plug-in's* para los diferentes programas de grabación, a través de los cuales se pudiera consultar en tiempo real el desfase entre las señales dispuestas en nuestro mezclador.
También podría considerarse la opción de establecer este tipo de *plug-in's* dentro de las mesas digitales para hacer la captación alineada desde un principio.

Apéndice A

Código del software

```
function varargout = MScorrelation_b(varargin)
% MSCORRELATION_B MATLAB code for MScorrelation_b.fig
%      MSCORRELATION_B, by itself, creates a new MSCORRELATION_B or↙
raises the existing
%      singleton*.
%
%      H = MSCORRELATION_B returns the handle to a new MSCORRELATION_B or↙
the handle to
%      the existing singleton*.
%
%      MSCORRELATION_B('CALLBACK',hObject,eventData,handles,...) calls↙
the local
%      function named CALLBACK in MSCORRELATION_B.M with the given input↙
arguments.
%
%      MSCORRELATION_B('Property','Value',...) creates a new↙
MSCORRELATION_B or raises the
%      existing singleton*.  Starting from the left, property value pairs↙
are
%      applied to the GUI before MScorrelation_b_OpeningFcn gets called.↙
An
%      unrecognized property name or invalid value makes property↙
application
%      stop.  All inputs are passed to MScorrelation_b_OpeningFcn via↙
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only↙
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MScorrelation_b

% Last Modified by GUIDE v2.5 07-Aug-2013 10:57:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MScorrelation_b_OpeningFcn, ...
                   'gui_OutputFcn',  @MScorrelation_b_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MScorrelation_b is made visible.
function MScorrelation_b_OpeningFcn(hObject, eventdata, handles,↙
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MScorrelation_b (see VARARGIN)
```

```
% Choose default command line output for MScorrelation_b
handles.output = hObject;



% ----------------INICIALIZACIÓN DE VARIABLES -------------------


%Creo e inicializo las variables necesarias utilizando estructuras↙
handles
%para poder intercambiar datos entre funciones sin tener que pasarlos en
%cada cabecera
handles.sample0_1 = 0; %fragmento 1 de Mic_0
handles.sample0_2 = 0; %fragmento 2 de Mic_0
handles.sample0_3 = 0; %fragmento 3 de Mic_0
handles.sample1_1 = 0; %fragmento 1 de Mic_1
handles.sample1_2 = 0; %fragmento 2 de Mic_1
handles.sample1_3 = 0; %fragmento 3 de Mic_1
handles.complete_sample_0 = 0;          %Toma completa de Mic_0
handles.complete_sample_delayed=0;      %Toma retrasada
handles.complete_audioplayer_sample_0=0;%Objeto para reproducir Mic_0
handles.complete_sample_1 = 0;          %Toma completa de Mic_0
handles.complete_audioplayer_sample_1 = 0;%Objeto para reproducir Mic_1
handles.delay = 0;   %Retraso hallado mediante correlación cruzada
handles.samples = 1; %Elección de escala "muestras"
handles.cm = 0;      %Elección de escala "centímetros"
handles.ms = 0;      %Elección de escala "milisegundos"
handles.sampleFreq = 44100; %Elección de frecuencia de muestreo
handles.numBits = 16;       %Elección de número de bits
handles.complete_sum = 0;                   %Suma de Mic_0 y Mic_1
handles.complete_sum_audioplayer_notalign=0;%Objeto para reproducir la↙
suma de Mic_0 y Mic_1
handles.complete_sum_align = 0;             %Suma de Mic_0 y Mic_1↙
alineada ajustando el retraso
handles.complete_sum_audioplayer_align=0;   %Objeto para reproducir suma↙
de Mic_0 y Mic_1 alineada ajustando el retraso
handles.Mic0 = 1;       %Elección de reproducción "Mic_0"
handles.Mic1 = 0;       %Elección de reproducción "Mic_1"
handles.NotAlign = 0;   %Elección de reproducción "suma de Mic_0 y Mic_1"
handles.Align = 0;      %Elección de reproducción "suma alineada de Mic_0↙
y Mic_1"
% guardas cambios en las estructuras handles;
guidata(hObject, handles);



% --- Outputs from this function are returned to the command line.
function varargout = MScorrelation_b_OutputFcn(hObject, eventdata,↙
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
%-------------ENTRADA DE DATOS-----------------------------


% --- Executes on button press in pushbutton1_close.
function pushbutton1_close_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1_close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

opc=questdlg('¿Do you want to exit?','EXIT','YES','NO','NO');
if strcmp(opc,'NO')
return;
end
clear,clc,close all;
close (gcbf);



% --- Executes on button press in pushbutton3_Mic0.
function pushbutton3_Mic0_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3_Mic0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[file, path] = uigetfile(('*.wav'),'Microphonic sample to process');
filewithpath = strcat(path, file);
[handles.complete_sample_0, Fs, nbits] = wavread(filewithpath);

if Fs ~= handles.sampleFreq || nbits ~= handles.numBits || length↙
(handles.complete_sample_0) < 12*Fs

   errordlg('The sample has not the specificed format', 'ERROR');
   pause(4);
   [file, path] = uigetfile(('*.wav'),'Microphonic sample to process');
   filewithpath = strcat(path, file);
   [handles.complete_sample_0, Fs, nbits] = wavread(filewithpath);

end

set(handles.text7_Mic0_Path,'String',num2str(filewithpath));
handles.sample0_1 = wavread(filewithpath, [Fs 2*Fs]);
handles.sample0_2 = wavread(filewithpath, [5*Fs 6*Fs]);
handles.sample0_3 = wavread(filewithpath, [10*Fs 11*Fs]);
guidata(hObject, handles);


% --- Executes on button press in pushbutton4_Mic1.
function pushbutton4_Mic1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4_Mic1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[file, path] = uigetfile(('*.wav'),'Microphonic sample to process');
filewithpath = strcat(path, file);
[handles.complete_sample_1, Fs, nbits] = wavread(filewithpath);

if Fs ~= handles.sampleFreq || nbits ~= handles.numBits || length↙
(handles.complete_sample_1) < 12*Fs

    errordlg('The sample has not the specificed format', 'ERROR');
    pause(4);
    [file, path] = uigetfile(('*.wav'),'Microphonic sample to process');
    filewithpath = strcat(path, file);
    [handles.complete_sample_1, Fs, nbits] = wavread(filewithpath);
```

```
end

set(handles.text8_Mic1_Path,'String',num2str(filewithpath));
handles.sample1_1 = wavread(filewithpath, [Fs 2*Fs]);
handles.sample1_2 = wavread(filewithpath, [5*Fs 6*Fs]);
handles.sample1_3 = wavread(filewithpath, [10*Fs 11*Fs]);
guidata(hObject, handles);


%-----------------CÁLCUL DEL DESFASE ENTRE SEÑALES-------------------


% --- Executes when selected object is changed in uipanel1_scale.
function uipanel1_scale_SelectionChangeFcn(hObject, eventdata, handles)
%   hObject:  handle to the selected object in uipanel1_scale
%   eventdata:structure with the following fields (see UIBUTTONGROUP)
%   EventName:string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty
%             if none was selected
%   NewValue: handle of the currently selected object
%   handles:  structure with handles and user data (see GUIDATA)

switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.
    case 'radiobutton1_samples'
        % Code when radiobutton1_samples is selected.
        handles.samples=1;
        handles.cm=0;
        handles.ms=0;
        guidata(hObject, handles);
    case 'radiobutton2_cm'
        % Code when radiobutton2_cm is selected.
        handles.samples=0;
        handles.cm=1;
        handles.ms=0;
        guidata(hObject, handles);
    case 'radiobutton3_ms'
        % Code when radiobutton3 is selected
        handles.samples=0;
        handles.cm=0;
        handles.ms=1;
        guidata(hObject, handles);
    % Continue with more cases as necessary.
    otherwise
        % Code for when there is no match.
end


% --- Executes on button press in pushbutton5_calculate.
function pushbutton5_calculate_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5_calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

msgbox('Please wait while is been processed','ANALYSING');

%Creo el objeto Hxcorr que luego utilizaré con step para hallar la
%correlación
Hxcorr = dsp.Crosscorrelator;

%guardo en la variable local audio_cor_1 la correlación computada con↙
step
%entre la primera muestra de cada micrófono (sample0_1 y sample1_1)
audio_cor_1 = step(Hxcorr,handles.sample0_1,handles.sample1_1);
[V I]= max(audio_cor_1);
```

```
delay_1=I - length(handles.sample0_1);

%guardo en la variable local audio_cor_2 la correlación computada con↙
step
%entre la segunda muestra de cada micrófono (sample0_2 y sample1_2)
audio_cor_2 = step(Hxcorr,handles.sample0_2,handles.sample1_2);
[V I]= max(audio_cor_2);
delay_2=I - length(handles.sample0_2);

%guardo en la variable local audio_cor_3 la correlación computada con↙
step
%entre la tercera muestra de cada micrófono (sample0_3 y sample1_3)
audio_cor_3 = step(Hxcorr,handles.sample0_3,handles.sample1_3);
[V I]= max(audio_cor_3);
delay_3=I - length(handles.sample0_3);

%Hallo la media de los retrasos de cada fragmento
delay_0= [delay_1 delay_2 delay_3];
delay=round (mean(delay_0));
handles.delay = delay;

guidata(hObject, handles);

if handles.samples == 1
    %Show the result in samples
    set(handles.text2_result,'String',num2str(delay));
elseif handles.cm ==1
    %Show the result in centimetter
    t_s= (delay/44100);
    d_cm = (340*t_s)*100;
    set(handles.text2_result,'String',num2str(d_cm));
elseif handles.ms ==1
    %Show the result in miliseconds
    t_ms= (delay/44100)* 1000;
    set(handles.text2_result,'String',num2str(t_ms));
end

% Alineo Mic_0 o Mic_1 dependiendo del resultado de la correlación

if delay > 0 % Si resultado es positivo, significa que Mic_0 estará↙
restrasado "delay" muestras respecto de Mic_1
    %Aplico el retraso detectado sobre Mic_1, añadiendo al principio↙
tantos ceros como muestras esté
    %retrasada Mic_0
    Hdelay = dsp.Delay(abs(delay));
    handles.complete_sample_delayed= step(Hdelay,handles.↙
complete_sample_1);
    guidata(hObject, handles);

elseif delay < 0 %Si resultado es negativo, significa que Mic_1 estará↙
retrasado "delay" muestras respecto de Mic_0
    %Aplico el retraso detectado sobre Mic_0, añadiendo al principio↙
tantos ceros como muestras esté
    %retrasada Mic_1
    Hdelay = dsp.Delay(abs(delay));
    handles.complete_sample_delayed= step(Hdelay,handles.↙
complete_sample_0);
    guidata(hObject, handles);

end



% --- Executes on selection change in listbox2_Fs.
```

```
function listbox2_Fs_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2_Fs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function listbox2_Fs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2_Fs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns↙
called


% --- Executes on selection change in listbox3_nbits.
function listbox3_nbits_Callback(hObject, eventdata, handles)
% hObject    handle to listbox3_nbits (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function listbox3_nbits_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox3_nbits (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns↙
called




% ------------------LISTENING↙
MODULE----------------------------------------



% --- Executes on button press in pushbutton7_PLAY.
function pushbutton7_PLAY_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7_PLAY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.Mic0 == 1

    play (handles.complete_audioplayer_sample_0);

elseif handles.Mic1 ==1

    play (handles.complete_audioplayer_sample_1);

elseif handles.NotAlign ==1

    play (handles.complete_sum_audioplayer_notalign);

elseif handles.Align == 1;

    play (handles.complete_sum_audioplayer_align);


end

% --- Executes on button press in pushbutton8_PAUSE.
function pushbutton8_PAUSE_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8_PAUSE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

if handles.Mic0 == 1

    pause (handles.complete_audioplayer_sample_0);

elseif handles.Mic1 ==1

    pause (handles.complete_audioplayer_sample_1);

elseif handles.NotAlign ==1

    pause (handles.complete_sum_audioplayer_notalign);

elseif handles.Align == 1;

     pause (handles.complete_sum_audioplayer_align);
end


% --- Executes on button press in pushbutton9_STOP.
function pushbutton9_STOP_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9_STOP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.Mic0 == 1

    stop (handles.complete_audioplayer_sample_0);

elseif handles.Mic1 ==1

    stop (handles.complete_audioplayer_sample_1);

elseif handles.NotAlign ==1

    stop (handles.complete_sum_audioplayer_notalign);

elseif handles.Align == 1;

     stop (handles.complete_sum_audioplayer_align);

end


% --- Executes on button press in pushbutton11_RESUME.
function pushbutton11_RESUME_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11_RESUME (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.Mic0 == 1

    resume (handles.complete_audioplayer_sample_0);

elseif handles.Mic1 ==1

    resume (handles.complete_audioplayer_sample_1);

elseif handles.NotAlign ==1

    resume (handles.complete_sum_audioplayer_notalign);

elseif handles.Align == 1;

     resume (handles.complete_sum_audioplayer_align);
```

```
end



% --- Executes when uipanel19_Channels is resized.
function uipanel19_Channels_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to uipanel19_Channels (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes when selected object is changed in uipanel9_Channels.
function uipanel19_Channels_SelectionChangeFcn(hObject, eventdata,↙
handles)
% hObject    handle to the selected object in uipanel1_scale
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none↙
was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.
    case 'radiobutton13_Mic0'
        % Code for when radiobutton1_samples is selected.
        handles.Mic0=1;
        handles.Mic1=0;
        handles.NotAlign=0;
        handles.Align=0;
        handles.complete_audioplayer_sample_0 = audioplayer (handles.↙
complete_sample_0, handles.sampleFreq);
        guidata(hObject, handles);

    case 'radiobutton14_Mic1'
        % Code for when radiobutton2_cm is selected.
        handles.Mic0=0;
        handles.Mic1=1;
        handles.NotAlign=0;
        handles.Align=0;
        handles.complete_audioplayer_sample_1 = audioplayer (handles.↙
complete_sample_1, handles.sampleFreq);
        guidata(hObject, handles);

    case 'radiobutton15_Notaligned'
        % Code for when radiobutton3 is selected
        handles.Mic0=0;
        handles.Mic1=0;
        handles.NotAlign=1;
        handles.Align=0;
        handles.complete_sum = handles.complete_sample_0 + handles.↙
complete_sample_1;
        handles.complete_sum_audioplayer_notalign = audioplayer (handles.↙
complete_sum, handles.sampleFreq);
        guidata(hObject, handles);


    case 'radiobutton16_Aligned'
        % Code for when radiobutton2_cm is selected.
        handles.Mic0=0;
        handles.Mic1=0;
        handles.NotAlign=0;
        handles.Align=1;
        if handles.delay > 0
            handles.complete_sum_align = handles.complete_sample_0 +↙
handles.complete_sample_delayed;
```

```
            handles.complete_sum_audioplayer_align = audioplayer↙
(handles.complete_sum_align, handles.sampleFreq);
            guidata(hObject, handles);

        elseif handles.delay < 0
            handles.complete_sum_align = handles.complete_sample_1 +↙
handles.complete_sample_delayed;
            handles.complete_sum_audioplayer_align = audioplayer↙
(handles.complete_sum_align, handles.sampleFreq);
            guidata(hObject, handles);

        end
        guidata(hObject, handles);

    % Continue with more cases as necessary.
    otherwise
        % Code for when there is no match.

end


% --- Executes during object creation, after setting all properties.
function uipanel19_Channels_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel19_Channels (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns↙
called



%-------------------SAVE MODULE-------------------------------


% --- Executes on button press in pushbutton12_ALIGNED.
function pushbutton12_ALIGNED_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12_ALIGNED (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.delay > 0
    [file, path] = uiputfile(('Mic_1_delayed.wav'),'Save Mic 1 aligned↙
sample as ');
    filewithpath = strcat(path, file);
    wavwrite(handles.complete_sample_delayed, handles.sampleFreq,↙
handles.numBits, filewithpath);
    set(handles.text9_aligned_Path,'String',num2str(filewithpath));
    guidata(hObject, handles);

elseif handles.delay < 0
    [file, path] = uiputfile(('Mic_0_delayed.wav'),'Save Mic 0 aligned↙
sample as ');
    filewithpath = strcat(path, file);
    wavwrite(handles.complete_sample_delayed, handles.sampleFreq,↙
handles.numBits, filewithpath);
    set(handles.text9_aligned_Path,'String',num2str(filewithpath));
    guidata(hObject, handles);


end


% --- Executes on button press in pushbutton14_ALIGNED_SUM.
function pushbutton14_ALIGNED_SUM_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14_ALIGNED_SUM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
[file, path] = uiputfile(('aligned_sum_of_audio_signals.wav'),'Save↙
aligned sum as ');
filewithpath = strcat(path, file);
wavwrite(handles.complete_sum_align, handles.sampleFreq, handles.numBits,↙
filewithpath);
set(handles.text10_aligned_sum_Path,'String',num2str(filewithpath));
guidata(hObject, handles);



% --- Executes on button press in pushbutton15_NOT_ALIGNED_SUM.
function pushbutton15_NOT_ALIGNED_SUM_Callback(hObject, eventdata,↙
handles)
% hObject    handle to pushbutton15_NOT_ALIGNED_SUM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[file, path] = uiputfile(('sum_of_audio_signals.wav'),'Save normal sum as↙
');
filewithpath = strcat(path, file);
wavwrite(handles.complete_sum, handles.sampleFreq, handles.numBits,↙
filewithpath);
set(handles.text11_not_aligned_Path,'String',num2str(filewithpath));
guidata(hObject, handles);
```

Bibliografía

[1] Christopher L. Stone. *Orquestal Recordings Techniques*. Audio Impressions, 2005.

[2] Matlab, MathWorks Inc, R2011b (7.13.0.564). *dsp.croscorrelation class*, 2011.

[3] Chris Cannam and Queen Mary, University of London, Copyright 2006-2011, creative commons licens. *Sonic Visualiser*, 2013.

[4] Dominic Mazzoni, GNU GENERAL PUBLIC LICENS. *Audacity*, 2013.

[5] Et Al P. Filipi. *Acoustics - Basic Physics, Theory And Methods*. Elsevier, 1999.

[6] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1999.

[7] John Eargle. *The Microphone Book*. Elsevier, 2005.

[8] Austin R. Frey Lawrence E. Kinsler. *Fundamentals of Acoustics*. John Wiley And Sons, 1962.

[9] John Watkinson. *Introduction to Digital Audio*. Focal Press, 1994.

[10] F. Alton Everest. *The Master Handbook of Acoustic*. McGraw-Hill, 2001.

[11] Jenny Bartlett Bruce Bartlett. *On-Location Recording Techniques*. Focal Press, 1999.

[12] Leo L. Beranek. *Acoustics*. Acoustical Society of America, 1993.

[13] Bobby Owsinski. *The Mixing Engineer's Handbook*. MixBooks, 1999.

[14] Bobby Owsinski. *The Recording Engineer's Handbook*. Course Technology, 2009.

[15] Shure. *Microphone Techniques*. Shure Educational Publication, 2003.

[16] blackwell. Phase demystified. *soundonsound*, april(40):5, 2008.

[17] Jacob Benesty. Spatial correlation and time delay estimation. Technical report, Bell Laboratories, Lucent Technologies, February 2001.

[18] Helena Alves. Introducción a matlab: Audio. Technical report, Estudios Fónicos, Programa oficial de Postgrado, February 2011.

[19] Antonio Quintero. Tutorial bàsico para el manejo de señales con matlab. Technical report, Universidad Nacional de Quilmes, November 2004.

[20] International Journal of Computer Science and Network Security. *A GCC Time Delay Estimation Algorithm Based on Wavelet Transform*, May 2010.

[21] 113th ICAES, Los Angeles. *The Significance of Interchannel Correlation, Phase and Amplitude Differences on Multichannel Microphone Techniques*, October 2002.

[22] 114th ICAES, Amsterdam. *Microphone Technique Theory for Stereo and Surround*, march 2003.

[23] 118th ICAES, Barcelona. *A New Microphone Technique for Five-Channel Recording*, may 2005.

[24] International Workshop on Acoustic Echo and Noise Control. *TIME DELAY ESTIMATION USING SPATIAL CORRELATION TECHNIQUES*, september 2003.